CRANFIELD UNIVERSITY

HELEN L LOCKETT

A KNOWLEDGE BASED MANUFACTURING ADVISOR FOR CAD

SCHOOL OF ENGINEERING

PhD THESIS

CRANFIELD UNIVERSITY

SCHOOL OF ENGINEERING

PhD THESIS

Academic Years: 1998 - 2005

HELEN L LOCKETT

A KNOWLEDGE BASED MANUFACTURING ADVISOR FOR CAD

SUPERVISOR: DR M D GUENOV

July 2005

ABSTRACT

This thesis presents a knowledge based manufacturing advisor for Computer Aided Design. The aim of the project has been to develop techniques that can help designers to evaluate the manufacturability of moulded parts during the design process.

One of the major achievements in the research has been the development of a novel feature recognition approach to allow moulding features to be recognised from a CAD model. Existing feature recognition techniques are not appropriate for moulded parts and a new feature recognition approach has been developed that uses a mid-surface abstraction from the CAD solid model as the basis for feature recognition. The feature recognition methodology presented in this research is a graph based technique that uses an attributed mid-surface adjacency graph as the basis for feature recognition. Feature recognition algorithms have been developed for a range of common moulding features.

Two evaluation techniques are presented that measure the quality of the feature recognition results and provide a confidence measure for the manufacturing advice that is generated. The first evaluation technique uses the Hausdorff distance to measure the similarity of the mid-surface model to the CAD solid model that was used to generate it, and the second uses a face mapping technique to evaluate the completeness of the feature recognition results.

A demonstrator for the knowledge based manufacturing advisor has been developed which incorporates the feature recognition software and an expert system for moulding advice. The expert system provides an interactive software environment in which the user can input details about their part design, and receive tailored manufacturing advice for a range of moulding processes. The demonstrator has been tested on a range of realistic moulded parts.

ACKNOWLEDGEMENTS

I would like to thank the School of Engineering for supporting my PhD studies, and allowing me to undertake a PhD alongside my job at Cranfield University. I would also like to thank the Worshipful Company of Coach and Coach Harness Makers for their generous bursary.

Thank you to my supervisor Dr Marin Guenov for his advice during the project. His comments throughout this research have helped me to challenge my ideas and justify my reasoning. Thank you also to Jonathan Barclay at Ticona for taking an interest in the project and providing feedback from an industrial viewpoint.

Most of all I want to thank my husband Russel for his belief in my abilities, and my family for their love and support. The PhD is finished, and now I can think about the future!

Thought…had let down its line into the stream. It swayed, minute after minute, hither and thither among the reflections and the weeds, letting the water lift it and sink it until – you know the little tug – the sudden conglomeration of an idea at the end of one's line: and then the cautious hauling of it in, and the careful laying of it out? Alas, laid on the grass how small , how insignificant this thought of mine looked; the sort of fish that a good fisherman puts back into the water so that it may grow fatter and be one day worth cooking and eating…

But however small it was, it had, nevertheless, the mysterious property of its kind – put back into mind, it became at once very exciting, and important; and as it darted and sank, and flashed hither and thither, set up such a wash and tumult of ideas that it was impossible to sit still.

Virginia Woolf, A Room of One's Own, 1929.

TABLE OF CONTENTS

TABLE OF FIGURES

TABLE OF TABLES

ABBREVIATIONS

AAG	Attributed Adjacency Graph
API	Application Programming Interface
AWK	Pattern Scanning Language (Aho, Weinberger and Kernighan)
AMAG	Attributed Mid-surface adjacency graph
B-rep	Boundary Representation
CAD	Computer Aided Design
CLIPS	Knowledge Based System Shell (C Language Production System)
CNC	Computer Numerical Control
EAAG	Extended Attributed Adjacency Graph
FAG	Face adjacency graph
FRF	Feature recognition results factor
GNU	GNU's Not UNIX (free software foundation)
KBS	Knowledge Based System
MAG	Mid-surface adjacency graph
MQF	Mid-surface quality factor
NIST	National Institute of Standards and Technology
SCL	STEP Class Libraries
STEP	STandard for the Exchange of Product data
STL	Stereo-lithography

1 INTRODUCTION

In today's manufacturing industry there is continuous pressure to drive down costs and increase quality. The high level of competition in global manufacturing means that companies must continuously improve their product development processes, and product designs are analysed and optimised from an early stage to ensure that they meet their functional and aesthetic requirements at minimum cost.

Design for manufacture is an important part of the product development process because it ensures that the manufacturing constraints of a product are taken into account from an early stage in the process. Historically, manufacturing engineers were not involved in product development until late in the design process, which meant that it was often too late for them to influence design decisions that might have a major impact on manufacturing feasibility or cost.

Effective design for manufacture requires that design engineers have extensive knowledge of the capabilities of available manufacturing processes and materials, and have an understanding of the design requirements for those processes.

A major difficulty in implementing effective design for manufacture is ensuring that design engineers have access to the relevant information and knowledge in a format that will allow them to easily perform design for manufacturability evaluations. To some extent the problem can be resolved through improved communications between design and manufacturing departments, but these communications are made more difficult by the trend within large manufacturing organisations to outsource many of their manufacturing operations, or undertake their design and manufacture operations at different geographic locations.

The aim of this project is to develop techniques that support design for manufacture for moulding processes. The project is focussed on plastic injection moulding, and metal casting which have been selected for study because they make significant demands on the designer, and represent a group of processes from which a designer may need to select the most appropriate solution. The objective of the project is to develop a software tool that can be integrated with the designers CAD system to provide manufacturing advice that is tailored to the current design.

1.1 Background

Design for manufacture is particularly important for moulded parts because the cost and quality of parts that are manufactured using moulding processes is highly geometry dependent. Moulding processes such as injection moulding and casting are extremely flexible and can be used to manufacture products with a wide range of styled or aerodynamic curved shapes. However, products that are manufactured using these processes must be designed with regard to the constraints of the manufacturing process to ensure that the part can be made to acceptable cost and quality.

The limitations of moulding processes are mostly due to the behaviour of the molten material as it fills and cools in the mould. The mould filling and mould cooling requirements make specific demands on the designer and a badly designed part may be impossible to manufacture, or may only be manufacturable with unacceptable cost or quality.

The following sections provide some background on the manufacturing processes that have been studied in this research, and the design for manufacture requirements for each process.

1.1.1 Injection Moulding

Injection moulding is used extensively to manufacture components for a wide range of products including electronics devices, domestic appliances, automotive interiors, and many others. Injection moulding allows parts with complicated shapes to be mass produced economically, and with very little waste. It is mostly used to manufacture plastic parts using thermoplastics or thermosets, but has also been applied to fine metal powders (Metal Injection Moulding or MIM).

In the injection moulding process heated molten plastic is forced into a mould under high pressure using a screw mechanism (Boothroyd, Dewhurst and Knight, 2002). The mould is usually heated or cooled to ensure that the part filling and cooling occurs evenly to aid the plastic solidification. When the part has cooled sufficiently the mould is opened and the part is ejected from the mould. The process is then repeated for the next part.

The set up costs for an injection moulded part are very high. Moulding machines are expensive to purchase, and the tooling costs are high which makes injection moulding only economic for mass production. A key factor in the cost effectiveness of a moulding production run is the cooling time for the part in the mould. The cooling time is usually the main factor in the total moulding cycle time, and can therefore have a major impact on the part cost. For thermoplastics the cooling time increases proportionally to the square of the wall thickness, meaning that a small increase in wall thickness can have a major impact on part cost (Boothroyd, Dewhurst and Knight, 2002).

An important factor in the achieving a high quality manufactured part is to minimise variations in wall thickness on the part. Thick or heavy sections can cause uneven cooling, and cause warping or appearance defects in the finished part. Injection moulded parts must therefore be designed with a thin and relatively constant wall thickness.

Design for injection-moulding guidelines are available from a number of different sources. There are standard design handbooks such as Bralla (1986) and Boothroyd, Dewhurst and Knight (2002) that provide manufacturing guidelines for a range of manufacturing processes. Injection moulding material suppliers also provide detailed design guidelines for the materials that they supply, and their design handbooks can provide a valuable resource for designers (Ticona, 2000 and GE Plastics, 1997).

The following examples of general design for mouldability considerations are extracted from the GE Plastics Design Guide (GE Plastics, 1997, pp48 – 49). Note that the guidelines are often very general in nature, providing the designer with general advice as opposed to specific design parameters.

- "Ideally the nominal wall thickness is kept constant due to shrinkage and cooling related issues"
- "In most applications, a thin, uniform wall with ribs is preferred to a thick wall"
- "Where changes in thickness are involved, care should be taken that the direction of melt flow during the moulding process is always from a thick area into a thinner section"
- "Wall thickness variation influences cooling rates of the moulded component and unequal thickness causes an imbalance of cooling… which can result in warping and appearance defects."
- "In order to reduce sink marks on prime appearance surfaces, the base thickness of the rib should not exceed 50% of the adjoining wall thickness."

(GE Plastics, 1997, pp48 – 49)

1.1.2 Casting

Casting processes are used to manufacture metallic parts with a wide range of shapes. All casting processes use the same general principle of pouring molten metal into a mould to form a metallic part, but there are a wide range of different casting processes. Three major casting processes are described here (sand-casting, investment-casting and die-casting).

Sand-casting uses tempered sand to form a sacrificial mould. The moist sand is pressed into wood or metal pattern halves, the patterns are then removed, and the mould is assembled. Molten metal is poured into the mould and left to cool, and the mould is broken to remove the cast parts. Sand casting can be used to manufacture parts ranging from approximately 30 grams to 200 tonnes, and can achieve a maximum tolerance of ±0.6 mm across a 25 mm diameter part. (Bralla,1986).

Investment casting uses a pattern made from wax or plastic. The patterns are assembled and then coated in fine grained slurry which dries to form a mould with a fine surface texture. The coated patterns are then heated to melt the wax or plastic and it is allowed to drain from the mould. Molten metal is then poured into the mould, and the mould is broken to remove the parts. Investment casting is typically used to manufacture parts up to approximately 5 kg in weight, and predominately for parts weighing less than 0.6 kg. The maximum tolerance that can be achieved is ±0.15 mm for a 25 mm diameter part. (Beeley, 2001).

In die-casting the mould is manufactured from hardened steel and the moulded metal is injected into the mould under high pressure. The part is cooled and the dies are opened to eject the part in a similar way to injection-moulding. Die casting is typically used to manufacture parts that are a maximum of 20 – 40kg, due to the limitations in machine size. The maximum tolerance that can be achieved is ±0.076mm for a 25 mm diameter part. (Bralla, 1986, Beeley, 2001).

The various casting processes are able to produce parts to a range of different sizes and tolerances. Sand casting is in general a cheaper and less precise process than investment or die casting, but it can be used for larger parts. Die casting is the most accurate, but the set up costs are high, and it is only cost effective for longer production runs.

All casting processes require that the metal "section [thickness] is normally kept as light as possible consistent with the required strength and rigidity and with metal flow" (Beeley, 2001). Cast parts should generally be designed to have a uniform section, or if a section change is required a progressive section change should be used. Isolated heavy sections should be avoided. Figure 1.1 shows examples of guidelines for the design of casting intersections on sand cast parts (Bralla, 1986).

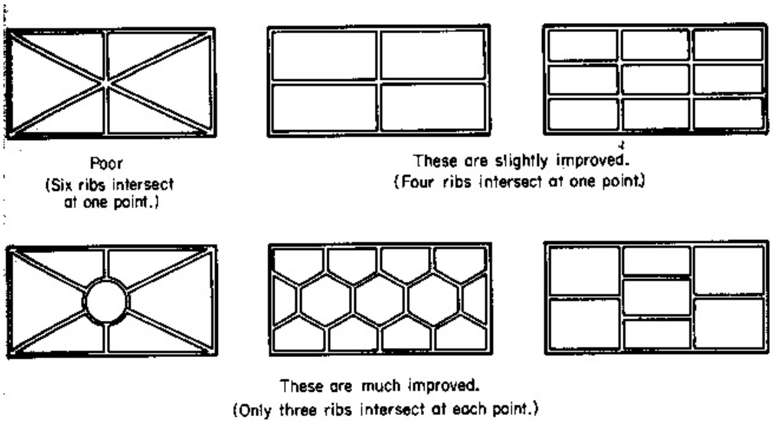

Poor
(Six ribs intersect at one point.)
These are slightly improved.
(Four ribs intersect at one point.)
These are much improved.
(Only three ribs intersect at each point.)

FIG. 5.1-9 Reduce the number of reinforcing ribs which intersect at one point so as to reduce the necessary thickness at the intersection. This helps prevent voids at the intersection.

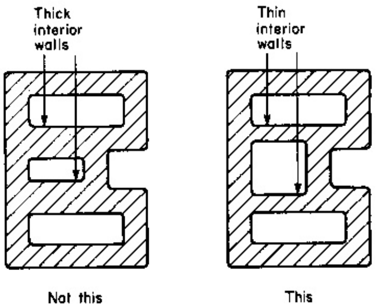


FIG. 5.1-17 Interior walls should be 20 percent thinner than exterior walls since they cool more slowly.

Figure 1.1 Example design guidelines for Sand Cast parts (Bralla, 1986)

1.1.3 Summary

It can be seen from the preceding sections that although moulding processes are extremely flexible, they also make some specific demands on the designer. The designer of a moulded part must take into account the mould filling and cooling behaviour of the part from an early stage in the design process. The moulding processes that have been described in this chapter have some common requirements, and each has their own specific design rules. A common requirement for all the moulding processes that have bee presented is that parts must be designed with a thin and relatively constant wall thickness.

Traditionally design for moulding guidelines have been available from design handbooks, and through communication with manufacturing experts at the mould maker or foundry. Although design for moulding information is available from a range of information sources, design for manufacture is often not considered in the early stages of the design process and design changes may be required to resolve manufacturing problems. The later in the design process that design changes occur the more expensive they are to resolve, and there is a significant benefit to be obtained by incorporating manufacturing issues from early in the product development process. (Boothroyd, Dewhurst and Knight, 2002).

1.2 Project Aims

The primary objective of this research has been to develop techniques that can help designers to incorporate design for manufacture requirements from an early stage in the design process. As previously described in Section 1.1 the design guidelines for moulded parts are well understood, but the difficulty is in delivering manufacturing advice to the designer in a way that will allow them to apply the advice to their own design. The availability of relevant and guided manufacturing advice is particularly important for inexperienced designers, or designers who are designing parts for an unfamiliar manufacturing process.

The approach that has been followed in this research has been to develop methodologies that can be implemented in a computer software tool to aid designers working in a CAD system. The main elements of the research have been the development of a novel feature recognition methodology to intelligently describe the shape of a moulded part, evaluation techniques to measure the confidence with which results can be used and a knowledge based system to provide moulding advice to the designer. These main elements of the project are outlined in more detail below:

1.2.1 Feature Recognition

A feature recognition approach for moulded parts is required to allow the manufacturing advisor to be integrated with geometry from a CAD system. Features that are of interest for manufacturability evaluation need to be identified from the CAD model and used as inputs to the manufacturing advisor. The feature recognition approach must be able to recognise common moulding features from parts with a wide range of complicated geometric shapes.

1.2.2 Evaluation tools

Evaluation tools will be required to allow the designer to judge the quality of the feature recognition results, and the confidence with which the advice from the knowledge base can be used.

1.2.3 Knowledge Base

A knowledge base will be developed to provide manufacturing advice for a range of moulding processes. The knowledge base needs to be structured so that the manufacturing advice can be generated at different levels of detail appropriate to different stages in the design process.

1.2.4 Novelty

The novelty in the research is in the development of a new feature recognition methodology that is applicable to thin-walled moulded parts, and the development of supporting evaluation techniques to test the quality of the feature recognition results. The research will also develop a demonstrator for an integrated manufacturing advisor tool that integrates a knowledge based system with a CAD system to provide tailored manufacturing advice to a designer using a CAD system.

1.3 Project Scope

The project scope has been limited to design for manufacture for moulding processes that require thin walled part geometries (specifically injection moulding and casting). The focus of the project has been the development of techniques to recognise moulding features from a CAD model, and to generate manufacturing advice for those features.

Design for moulding is a large research area that encompasses part design, mould die and runner design, structural and thermal analysis, materials behaviour and many other aspects. This project has focussed on the design for moulding aspects that are of relevance to the product designer in the early stages of the product design process –

particularly those that have impact on the process selection, material selection and geometric shape of the designed part. The aim of the research is to develop techniques that are complementary to existing analytical tools for mould filling and mould cooling analysis, by helping designers to make their initial design closer to a manufacturable solution.

The feature recognition approach that has been developed is applicable only to thin walled parts, and cannot be applied to part geometries with very thick or chunky regions, but this limitation is well aligned with the requirements of injection-moulding and casting processes. Future work could extend the scope of the feature recognition technique to other thin walled manufacturing processes such as sheet metal design, or possibly composite part design.

1.4 Thesis Structure

Chapter 2 provides a critical review of the relevant literature and introduces the main research areas in the thesis. Chapter 3 describes the feature recognition methodology that has been developed, and the feature classification scheme for moulded parts. In Chapter 4 the feature recognition evaluation techniques are presented which provide a confidence measure for the feature recognition results. Chapter 5 presents the manufacturing advisor architecture, and the information gathering process. Chapter 6 describes the implementation of a demonstrator to test the main methodologies that have been developed, and Chapter 7 describes the evaluation and testing for some real world case studies. In Chapter 8 the results are discussed, and in chapter nine conclusions are drawn.

2 LITERATURE REVIEW

2.1 Introduction

This chapter provides a critical review of the relevant literature for this project. The review evaluates existing research in the various elements of the manufacturing advisor including CAD Feature recognition, the use of mid-surface models, evaluation tools, and also existing manufacturing advisor research projects.

2.2 Feature Recognition

Over the last 20 years there has been a great deal of interest in the development of computer tools that can identify features with real world meaning from 2D images or 3D design models. Shah and Mantyla define a feature as follows:

> *"Features are generic shapes or characteristics of a product with which engineers associate certain attributes and knowledge useful to reasoning about that product. Features encapsulate the engineering significance of portions of the product geometry." (Shah and Mantyla,1995)*

The term feature recognition refers to techniques that are able to automatically identify features from a product geometry for some manufacturing or other purpose. Feature recognition has been widely studied in image recognition: for example optical character recognition for recognising text from scanned documents and facial feature recognition for security systems; and in Computer Aided Design: largely for manufacturing applications.

Most CAD feature recognition techniques are based on the concept of parsing geometry and topology entities in a CAD database to identify patterns or regions that represent features of interest. Han, Pratt and Regli (2000) published a review of the status of feature recognition from solid models and identified three main areas of active research in feature recognition: graph-based methods, volumetric decomposition methods and hint-based methods.

Graph based feature recognition translates the boundary representation of a solid model into a graph structure which is then parsed to find sub-graphs that match known feature patterns. Volumetric decomposition methods decompose an object into a set of intermediate volumes which can then be processed to find features. Hint-based methods are a development of the other two techniques in which the goal is not to find exact feature matches, but to look for similar matches that can be used as “hints” for feature recognition. The following sections describe some key research projects in each of these areas, and highlight their limitations for this application.

2.2.1 Graph Based Methods

Joshi and Chang (1988) developed an early graph based feature recognition approach for machining features. In their work the feature recognition is performed using a graph structure that is built on the underlying B-rep CAD solid model. The graph structure is an attributed adjacency graph (AAG) in which the nodes of the graph represent the part faces and the arcs of the graph represent the edges. Each graph arc also stores an attribute to specify whether the edge is concave or convex. Features are recognised by searching for sub-graphs corresponding to predefined features in the attributed adjacency graph. Figure 2.1 shows a simple example of an AAG graph segment. In the figure a slot is represented by three faces of the solid model (F_1, F_2 and F_3). The graph segment for the slot shows that F_1 is connected to F_2 and F_2 is connected to F_3, and the attributes on the graph arcs show the F_1 forms a concave angle with F_2 and F_2 forms a concave angle with F_3. The slot feature can therefore by matching the graph segment from the part to the stored graph segment for a slot feature.

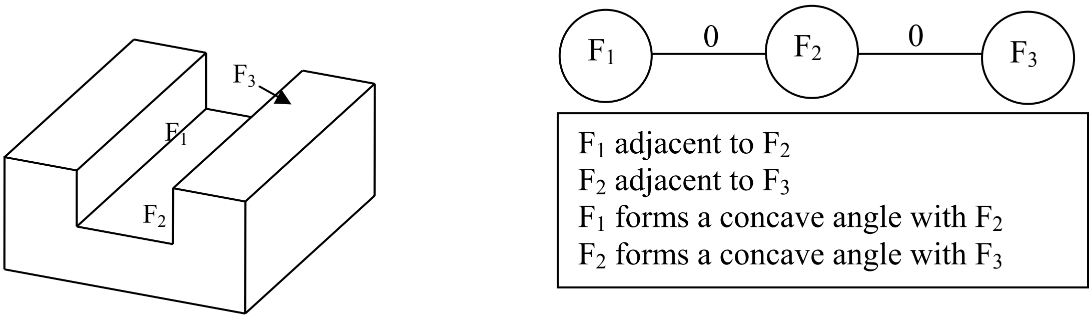


Figure 2.1 Feature Recognition Using Attributed Adjacency graph (AAG) (Joshi and Chang, 1988)

The work of Joshi and Chang has been further developed by a number of researchers including Gao and Shah (1998) who developed the extended attributed adjacency graph (EAAG) which captures several face and edge attributes in the graph. The EEAG graph structure stores a number of geometric attributes with each graph node and arc and provides additional information to aid the recognition of interacting features. They use a hint-based method to recognise and extract alternative feature interpretations. However, their work does not contribute to the recognition of features with sculptured surfaces.

2.2.2 Volume-decomposition Methods

Kim (1992) developed a convex decomposition method for feature recognition from solid models following on from earlier work by Kyprianou (1980). Kim uses a technique called alternating sum of volumes with partitioning (ASVP) to a B-rep solid model to recognise volumetric form features from the model. In this approach the feature recognition is performed by decomposing the solid part into a hierarchical structure of convex elements, where each element represents a volumetric feature on the part. The main problem with volume-decomposition methods is that the operations in each step do not guarantee success, and it may not be possible to generate a feasible model. This approach is also only applicable to polyhedral parts with planar faces, so curves have to be approximated as straight lines before processing.

2.2.3 Hint-based Methods

Hint based feature recognition may be based on either of the other main feature recognition approaches, but is characterised by searching for similar matches that may indicate the existence of a feature and not exact matches. This approach has benefits over other techniques when searching for intersecting features, because some faces of a feature may be removed by another intersecting feature which would usually prevent successful recognition. Vandenbrande and Requicha (1993) developed a hint based feature recognition approach that undertakes the feature recognition in four steps. Firstly feature hints are generated by searching for characteristic combinations of part faces on the solid model, secondly the feature hints are classified into promising, unpromising and rejected hints, thirdly the promising hints are processed by a "feature completer" to generate the actual features, and finally a verification step is performed to ensure that the recognised features are valid. Vandenbrande and Requicha claim that their approach is better at recognising interacting features that earlier work, but they are still limited to 2.5-D swept features, and mostly prismatic parts.

All three of the main approaches described above were developed for the recognition of machining features from prismatic solid models. All of the techniques experience problems with recognising intersecting and composite features. Researchers have continued to develop feature recognition techniques in recent years that have made some extensions to the field of application, or specific problems of feature interactions, but the overall limitations of CAD feature recognition remain.

In their review paper Han, Pratt and Regli (2000) state that there are still unsolved problems for feature recognition from machined parts (intersecting features, handling multiple interpretations, controlling computational complexity). They also identify that the vast majority of feature recognition work to date has focussed on machining features and assumes that the part is generated from a rectangular stock material. They state that

> "*Many more parts are manufactured by processes such as sheet metal stamping, die casting and injection moulding that are made by machining…There has been*

> *some feature-based work relating to most of these areas, but not very much. The field is wide open for new and valuable contributions in all of these areas".* (Han, Pratt and Regli, 2000)

2.2.4 Recent Developments in Feature Recognition

In recent years there have been several developments in feature recognition that relate to curved parts, and to injection moulding.

Zhang et al (2004) developed a surface based approach to feature recognition for freeform surface machining features. They are able to identify machining regions from a set of complex surface patches representing the faces to be machined. This approach has some similarities to the requirements for feature recognition from moulded parts because it supports complex curves surfaces, and its focus is on identifying surface regions, as opposed to volumetric regions. However, the feature types of interest in this research are very different to those for moulded parts.

El-Mehalawi and Miller (2003) describe a project to identify parts with similar geometric characteristics from a database of mechanical components. Although this research project is not specifically a feature recognition approach it faces some of the issues that are found in feature recognition. El-Mehalawi and Miller's research uses an Attributed Adjacency Graph (AAG) to represent the part geometry in the database, and captures geometric and topological characteristics from the graph to allow the database to be searched for similar parts. In this project the data exchange between the CAD system and the geometry graph is undertaken using the STEP standard. This project is applied to solid models of prismatic machined parts.

Chen, Wen and Ho (2003) developed a feature based design approach for injection moulded parts in which they are able to automatically extract characteristics for manufacturability assessment from the feature model. This work is not strictly feature recognition because it requires that the model be constructed using predefined design features, but it is of interest because it evaluates interactions between the defined features. They define spatial relationships between features such as is_in, adjacent_to and coplanar, and they can automatically recognise these relationships from a feature model. This research project is almost the only existing feature recognition project for moulded parts that has been found in the literature, however, it is not a complete feature recognition implementation, and it is limited to simple geometric shapes.

Yin, Ding and Xoing (2001 & 2004) present a virtual prototyping approach for moulded parts. In their research they have developed a volumetric feature recognition method to identify undercut features on a moulded part. They use a convex decomposition method to identify moulding features, and then identify different interpretations of any interacting features that are found. The research is relevant because it provides a feature recognition scheme for moulded parts, but the focus in the research is on identifying undercut features, and the features that are recognised are very similar to those used in machining feature recognition. It is also limited to prismatic parts and cannot be applied to parts with sculptured surfaces.

Belludi and Yip-Hoi (2002) have developed a feature recognition approach based on a stereo-lithography (STL) representation of a CAD solid model. Their approach uses feature recognition to automatically identify and clean errors from a facetted STL model. This work has some parallels with feature recognition from mid-surface models because the topological relationships between the facets in the STL model have some similarities to the topology of a mid-surface model. However, the area of application for the research is machining for prismatic parts, and moulded parts are not considered.

A comprehensive literature review for CAD feature recognition has found only a small number of CAD feature recognition techniques for moulded parts, and only one that provides limited support for free-form surfaces. CAD feature recognition is still an active area of research, with the recognition of feature interactions and curved surfaces still not fully resolved. The limited progress in feature recognition for moulded parts is perhaps due to the difficulty of applying existing machining feature recognition techniques to the complex geometries and feature interactions on moulded parts.

2.3 Mid-Surface Models

The lack of progress in the literature of feature recognition for moulding parts has lead the author to investigate other geometric representations that could be used to model a moulded part to facilitate feature recognition. Most moulding processes require that parts must be designed with walls that are thin and of relatively uniform thickness due to the behaviour of the molten material during mould filling and cooling, and this characteristic means that a mid-surface abstraction of the part geometry can be used to represent the important geometric characteristics of the part in a simpler model form.

Mid-surface representations are widely used to analyse the behaviour of moulded parts. For example finite element analysis of a thin walled part can be performed with significantly less computational cost if the part can be dimensionally reduced to a model represented by 2D shell elements instead of 3D solid elements (Armstrong, 1994). Several CAD systems and Finite Element analysis pre-processors provide functions to automatically or semi-automatically construct a mid-surface abstraction from a CAD solid model.

2.3.1 Mid Surface Generation Techniques

The medial axis transform is a mathematical technique which can be used to obtain the "skeleton" of a 2D shape, and provides and an associated radius function which gives the thickness of the profile around the skeleton. The medial axis transform was first proposed by Blum (Blum, 1967) and can be defined as the locus of the centre of an inscribed disc of maximal diameter as it rolls around an object interior (Donaghy, 2000). The associated radius function gives the radius of the inscribed circle at every point on the skeleton, and makes the original 2D object recoverable from the medial axis. There are now robust algorithms for calculating the medial axis transform for 2D objects, and they are finding application in many areas including image processing, finite element and CAD applications. Figure 2.2 shows an example of the medial axis for a simple 2D profile that has been generated using an implementation of the medial axis method developed by Ogniewicz (1996). One of the problems of the medial-axis approach is that the medial-axis has unwanted branches that need to be trimmed before the medial-axis is representative of the original shape.

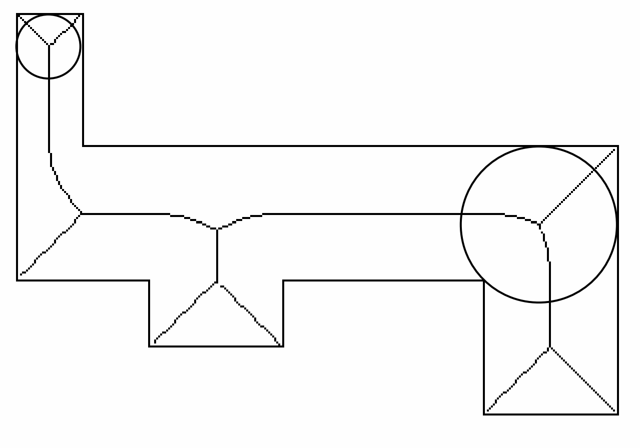


Figure 2.2 Example of Medial axis, and sample maximal circles. Created using the code developed by (Ogniewicz, 1996)

The medial surface transform is the three-dimensional equivalent of the medial axis transform that can be used to create the mid-surface of a three dimensional object. The medial surface is defined as the locus of the centre of a maximal sphere as it rolls around the object interior (Donaghy, 2000). The medial surface transform is a development from the two-dimensional medial axis transform developed by Blum and also incorporates a radius function to allow the original shape to be recovered from the medial surface. Algorithms to calculate the medial axis transform are reasonably mature, but the development of a robust algorithm for the medial surface transform has been slow because of the difficulty of computing the abstraction for a general solid object (Donaghy, 1996).

Rezayat (1996) describes a mid-surface algorithm based on surface pairing. In the surface pairing approach candidate surface pairs are identified on the solid model which represent thin walls on the part. A medial-surface is then constructed between each surface pair, and the resulting surfaces are trimmed and extended to form a consistent model. Rezayat claims that the surface-pairing approach has benefits over other medial-axis type techniques because the resultant geometry is cleaner and requires less reconstruction that those from medial axis approaches, however the surface pairing approach also has problems because it can be difficult to identify all of the surface pairs.

These algorithms have now been implemented in commercial software tools, which despite their limitations are able to generate mid-surfaces for a range of realistic designs. A medial surface toolkit is available from TranscenData (Transcendata, 1995), and the surface-pairing algorithm is implemented in UGS I-DEAS-NX (UGS, 2005).

2.3.2 Mid-surface Applications

Armstrong et al (Armstrong, 1994; Sheehy, 1996) use the medial axis transform for dimensional reduction in finite element modelling. In their work they identify features in a solid model that can be simplified by reducing their dimensionality, for example a long slender face may be reduced to a beam. Later work by the same group (Donaghy, 2000) uses the medial axis algorithm to reduce the dimensionality of a finite element model by generating the medial surface and medial axis for the part, and replacing the thinned-regions with thin shell or beam elements where possible.

Radhakrishnan, Amsalu et al (1996) use the medial axis transform as part of a design rule checker for sheet metal components. A key aspect of sheet metal design is checking the relative position of features on the sheet (e.g. the proximity of holes to bends or other features). Radhakrishnan and Amsalu use the medial axis transform to subdivide a sheet metal design into simple regions each containing one feature; they then apply manufacturing rules to the design to check that the relative positions of features on the part are acceptable. Dividing the design into sub-regions created using the medial axis significantly reduces the complexity of the search algorithms required to evaluate the relative positions of all the features.

Wozny, Pratt and Poli (1994) present a feature based design approach for early design. In their paper they present a mid-surface model representation and data structure, but their focus is on creating design models for the configuration stage of the design process and they do not attempt to recognise features from a mid-surface model, instead constructing a mid-surface model using a graph grammar.

Chu and Lee (1993) developed a thinning algorithm for design analysis of cast and forged parts. They used digital image processing techniques to perform a thinning operation on a voxel representation of a solid part, and then identify junctions on the skeleton that could cause forging or casting problems. The main limitations of their research are that they uses a voxel representation for the part geometry, which may make it difficult to obtain sufficient resolution for a complicated part and they only represent parts as one-dimensional (beam type) regions and do not consider thin-walled regions that are two-dimensional (shell type) regions.

More recent research projects have continued to use the mid-surface representation for analysis and simulation applications. Su, Lee and Senthil Kumar (2004) have developed a technique to link solid and shell element regions in a finite element mesh. They use an algorithmic approach to automatically define a mixed dimensional mesh on a non-manifold part.

Pao et al (2004) have developed a casting solidification simulation tool that uses the medial-surface of an object as the basis for simulation. They claim that their simplified casting evaluation can run “an order of magnitude” faster than a full 3D analysis, and the results compare well with a full analysis.

A comprehensive review of mid-surface applications has shown that mid-surface based techniques have been successfully applied to simplify engineering analysis of thin walled parts. In the majority of the research in this area the focus is on using the mid-surfaces for engineering analysis, or automatically identifying regions that can be represented by a mid-surface. No existing projects have been found in which feature recognition is performed on a mid-surface model, although Gadh, Gursoz, Hall, Prinz and Sudhalkar proposed the possibility of mid-surface based feature recognition (Gadh, Gursoz et al. 1991, Chern, Gursoz et al, 1990). In their research into feature abstraction they suggest a voxel based implementation of the medial-axis transform as a basis for feature recognition, but they state that “feature recognition capabilities have not been developed at this time”.

2.4 Evaluation Techniques For Feature Recognition

A final important aspect of feature recognition research is the requirement for techniques that can evaluate the quality of feature recognition results. The previous sections have stated that automatic feature recognition techniques are difficult to implement for real world parts, and it is therefore an essential to be able to judge how well a feature model represents the actual part geometry.

A wide review of feature recognition research has identified that most feature recognition approaches have been evaluated by testing them on a range of parts and reporting the success or otherwise of the results. The evaluation of feature recognition results does not appear to have been considered in the majority of feature recognition research papers, and where it is considered no conclusions are reached. For example Huang and Yip-Hoi (2002) identify the "problem of robustness of feature recognition technology", but they propose as a solution that "the use of feature recognition tools should be decided on a case by case basis and always supplemented by manual editing capabilities".

Feature recognition evaluation techniques can be considered in two main aspects – firstly measures of part complexity (which can be used to classify part geometry types prior to feature recognition), and secondly similarity measures that compare the feature recognition results against the geometry of the original CAD model.

2.4.1 Complexity Measures

Complexity measures can be used to classify parts and determine whether they are within the scope of a particular feature recognition algorithm. Several research groups have developed complexity measures for geometric parts that allow them to categorise part types for feature recognition, cost evaluation or other downstream manufacturing applications. Rodriguez-Toro et al (2002) present an approach for measuring shape complexity in support of assembly oriented design. They propose a shape complexity metric in which parts are classified using three base types (revolute, prismatic, or thin walled), and then ranked in complexity on a scale of 1 to 5 according to the number and types of features, and the complexity of the surfaces. Rodriguez-Toro proposes that a shape complexity measure could be used to compare shape similarity between components or as the basis for metrics to compare different types of complexity.

Little et al (1998) developed a feature complexity index (FCI) for 3D solid parts for the comparison of different feature recognition algorithms. The FCI can be used to compare how effectively different feature recognition algorithms are able to handle components of differing complexity. The FCI measures shape complexity in three separate areas – (i) type of geometry – planar, cylindrical, complex, combined (ii) the number of directions from which 2 1/2 D features may be observed (iii) the number of vertical face circuits for each orientation in (ii).

Yang et al (2003) propose a complexity measure for composite parts based on processing a stereolithography (STL) model of the part. In their approach they use the angles between adjacent triangles in the STL model to obtain a measure of complexity for the part by searching for regions of the part with rapid changes in shape as an indicator of part complexity.

2.4.2 Similarity Measures

Similarity measures can be used to compare the feature recognition results to the CAD geometry model used as input to the feature recogniser. Li and Liu (2002) present one feature recognition evaluation approach in their paper on feature recognition for the removal of detailed features from CAD models. They define a volume-simplification-ratio for the model which compares the volume of the simplified feature model with the volume of the original part to allow them to measure how much the part has been modified by the detail removal process.

In their review paper about three-dimensional shape searching Iyer et al (2005) present a range of similarity measures that could be applied to CAD and feature models. They identify the Minkowsky distance, Hausdorff distance and Correlation metric as metrics that can be used to measure the similarity of 3D shapes.

The review of evaluation techniques for feature recognition results has identified that although there are a range of complexity and similarity measures that could be used to assist with evaluating feature recognition results, these measures have not be widely applied to CAD feature recognition in past research projects.

2.5 Manufacturing Advisor Tools

The preceding sections of the literature review have focussed on the past research in the fields of feature recognition, mid-surface modelling and evaluation techniques which sets the scene for the main theoretical work in this thesis.

This final section of the literature review critically evaluates the existing manufacturing advisor tools that have been developed to assist with design for manufacture for moulded parts. These manufacturing advisor tools take a range of different approaches to generating manufacturing advice for a designer. The various tools are focused on different phases of the design process, and use a variety of different means to input design data into the tool. The common theme for all the tools is that they aim to encapsulate design knowledge to aid a designer with some aspect of moulded part development. Most manufacturing advisor tools use a knowledge based system or expert systems as a means to generate design advice.

Since the earliest developments in computing there has been interest in the development of "intelligent" machines. As early as 1963 Newell and Shaw (1963) developed a general problem solver to simulate human thought, and research into intelligent systems has continued until the present day. Meystel and Albus (2002, pp3) define intelligent systems as follows:

> *"Intelligence is the ability of a system to act appropriately in an uncertain environment, where an appropriate action is that which increases the probability of success and success is the achievement of behavioural subgoals that support the system's ultimate goal"*

Jackson (1999) defines an intelligent system as a computer program that in some way emulates human like reasoning on knowledge. A knowledge based system is a computer program which stores knowledge of a particular domain in a knowledge base and then applies reasoning techniques to solve problems or give advice about the domain. Facts in the knowledge base represent the knowledge in the system and rules are the instructions that determine how the facts can be combined to produce outputs (Jackson, 1999).

The manufacturing advisor tools that have been identified in the literature can be categorised into two main types – firstly knowledge based system based tools, and secondly analysis based tools.

2.5.1 Knowledge Based System Tools For Moulding Design

Tolga-Bozdana and Eyercioglu (2002) developed a frame-based modular expert system called EX-PIMM to determine injection moulding parameters and select an appropriate machine and material for a product. Their knowledge base has three modules – the first identifies a set of feasible moulding machines for a particular part and material, the second identifies feasible materials for the part and moulding machine, and the third combines the machine and material and determine the optimum number of cavities for the part, selected machine and material. The part design is input to the system using parameters for part volume, projected area and maximum dimension (length/ width/ height) of the part. Detailed design parameters and features are not considered.

Er and Dias (2000) describe a rule based expert system for casting process selection. Their system has five interconnected levels that take into account material selection, geometric factors, accuracy factors, production run size and cost. The first four levels are combined to narrow down the process selection, and the final costing level is used to provide a comparison between the remaining candidate processes. The expert system uses a question and answer dialog with the user to elicit the required system inputs. The geometric inputs to the system are the number of split planes required, number of holes and internal features, maximum and minimum section thickness, part length, and weight. The detailed geometry of the design is not considered.

Chin and Wong (1996) developed a knowledge based system for the evaluation of conceptual design development for injection moulded parts. Their prototype system EIMPPLAN-1 is able to select appropriate materials and generate major mould design features. The geometric inputs to the system are general part design parameters such as wall thickness, existence of undercuts and threads, number of parting lines, but not detailed design features.

A geometric mouldability analysis tool using fuzzy logic has been developed by Zhou-Ping and Han Ding (2004). Their objective is to achieve an optimal mould cavity design based on manufacturing and geometric considerations. Their tool is applied to the later stages of the design process and does not propose geometric design changes to the part.

None of the knowledge based manufacturing advisor tools that have been reviewed use the detailed part geometry to generate manufacturing advice.

2.5.2 Analysis Based Tools

The analysis based manufacturing advisor tools use geometric, manufacturing and other information to predict the behaviour of a moulded or cast part using analysis or simulation techniques. Preddy, Knight, Cowell and Mileman (1997) developed a design for casting system (CAST-AID) aimed at the manufacturing engineer in which the user builds a simplified representation of the part using a library of standard shapes (e.g. filleted L sections, T and cross junctions, bars wedges and plates). The modulus of the part is calculated by combining the known modulus for each of its simple constituent parts and the system is able to identify potential problems with mould filling and cooling. The designer can interactively manipulate the modulus gradient by adding taper, padding, chills and insulation to the model. Preddy's system is not integrated with a CAD system, but incorporates simple geometric representations of common moulding features.

Lu, Rebello, Miller and Kinzel (1995) use a voxel based approach to identify the geometry characteristics of a part (e.g. thick areas for hot-spot detection). In their system a 3D model of the part is subdivided into voxels (small cuboid volumes) and then the distance from every voxel to the part boundary is measured. The system is able to find variations in material thickness and identify potential hot spots in the casting. They assert that this approach is more flexible than feature recognition techniques, but the limitation of their system is that they are only able to evaluate the design for a particular aspect of manufacturability, and cannot extend the system to incorporate other aspects. The systems uses CAD geometry in the simulation, but does not perform feature recognition to interpret the geometry that is analysed.

Ravi and Srinivasan (1989) present a novel method for the identification of hot spots in complex 3D castings based on analysing the geometric shape of the part. 2D slices are taken through the part in orthogonal directions and the variations in thickness across each slice are plotted and combined to find heavy areas.

It can be seen that there have been a number of past research projects in the area of manufacturing advisor tools for moulded, cast and forged parts. These tools fall into two main types: knowledge based tools and analysis based tools. In the existing knowledge based tools the design information is input into the knowledge base by the user, and is very generic in nature, (wall thickness, number of features etc.); none of the tools that have been investigated have been integrated with a CAD database that contains the details design description of the part. In the analysis based tools the geometry is usually input using simplified geometry elements, and not linked to CAD data. The literature review has shown that there is a gap between the design tools that are used to model detailed components during the design process, and manufacturability evaluation tools that are used to aid a designer in designing for a particular manufacturing process.

2.6 Summary and Conclusions

This literature survey has reviewed the relevant background research for an intelligent manufacturing advisor using feature recognition. A comprehensive review of feature recognition research projects has identified that there has been very limited past research into developing feature recognition techniques that can be applied to moulded parts. One reason why researchers have not pursued feature recognition for moulded parts is that moulded parts are often composed of many feature interactions, and have complex curved surfaces and these are well known to cause difficulties for existing feature recognition techniques.

The use of mid-surface models to represent moulded parts for engineering analysis has also been reviewed. Mid-surface models are widely used to represent thin walled parts for finite element analysis, and can significantly reduce part complexity and computation time for analysis. However, the literature review has not identified any applications that use mid-surface models as the basis for feature recognition.

Robust feature recognition requires the development of techniques to allow the user to evaluate the quality of feature recognition results. Several techniques for shape comparison, and shape complexity measures have been identified in the literature, but these have not been applied to CAD feature recognition in the past.

Finally, although a range of moulding manufacturing advisor tools have been identified, none of them integrate a knowledge base with a CAD system. The missing link between Computer Aided Design and knowledge based tools can be achieved through the development of a feature recognition technique for thin walled moulded parts using a mid-surface approach. A common limitation of the projects that have been reviewed is that they are applicable only to a single manufacturing process, and a particular stage in the design process. A more flexible manufacturing advisor would be applicable to a range of processes, and at different stages in the design process.

3 FEATURE RECOGNITION

3.1 Introduction

This chapter describes a novel feature recognition methodology for moulded parts. Section 3.2 reviews the capabilities and limitations of existing feature recognition techniques in relation to moulded parts, Section 3.3 describes the basis for using mid-surfaces to represent the geometry of moulded part and Section 3.4 defines the graph structure that has been used to represent the mid-surface geometry. In Section 3.5 the feature classification is described and 3.6 presents the feature recognition approach. The feature recognition approach developed in this PhD research and described in this chapter has been published by the author (Lockett and Guenov, 2005). A copy of the publication is included as Appendix A.

3.2 Graph Based Feature Recognition

As already stated in the literature review CAD feature recognition is used to identify features from the geometry and topology information in a CAD geometry model for some manufacturing or other purpose. Feature recognition has received a great deal of research interest because it allows real world meaning to be applied to the geometry in a CAD model, but there are still some difficulties in developing useable feature recognition tools that can be applied to a wide range of practical parts. These difficulties include problems in identifying intersecting features, high computational complexity of feature recognition algorithms, and applicability to a limited range of geometric shapes (Shah and Mantyla, 1995 & Han, Pratt and Regli, 2000).

The majority of past research in CAD feature recognition has focused on the automation of process planning, NC programming and inspection planning for CNC machined parts (Shah and Mantyla, 1995). Feature recognition research has concentrated on recognising simple machined features such as cylindrical holes, slots, steps and pockets from prismatic parts. In machining feature recognition each feature represents a negative volume of material to be removed from the part in a single operation, and a common problem is to identify individual volumes from many possible combinations of intersecting features.

There are some particular difficulties in applying existing feature recognition techniques to moulded parts because the features of interest are different to those on machined parts. Moulded parts are designed with a thin and relatively thin main wall, to which additional thin walled features are attached to provide strength or other functionality. The objective of feature recognition for moulded parts is therefore to identify the thin walls in the part, and to find the characteristics of and intersections between those walls. A second difficulty in applying existing feature recognition techniques to moulded parts in that these parts tend to have complicated curved surfaces which cannot be evaluated using current methods.

The difficulty of applying a standard graph based feature recognition technique to moulding features can be illustrated with a simple example shown in Figure 3.1. Figure 3.1 (a) shows a model of a simple part with a rib feature attached to a main wall. Using a standard graph based feature recognition technique the rib could be identified by recognising the adjacency between faces F1, F2 and F3 and the convex angles between F1 and F2, and between F2 and F3. However, in Figure 3.1 (b) it can be seen that three similar faces to those in 3.1 (a) now form part of an X-junction. Using a standard graph based feature recognition approach it would be necessary to recognise two rib features (one formed by faces F1, F2 and F3 and the second formed by faces F4, F5 and F6) and then to search for a relationship between those two features to find the X-junction. Figure 3.1 (c) shows another example which is topologically similar to 3.1 (b), but which represents a different type of feature (the two rib features form a staggered T-junction instead of an X-junction). In order to differentiate between the features in 3.1 (b) and 3.1 (c) the feature recognition software would need to search for and evaluate all possible feature combinations in the model which would be extremely computationally expensive.

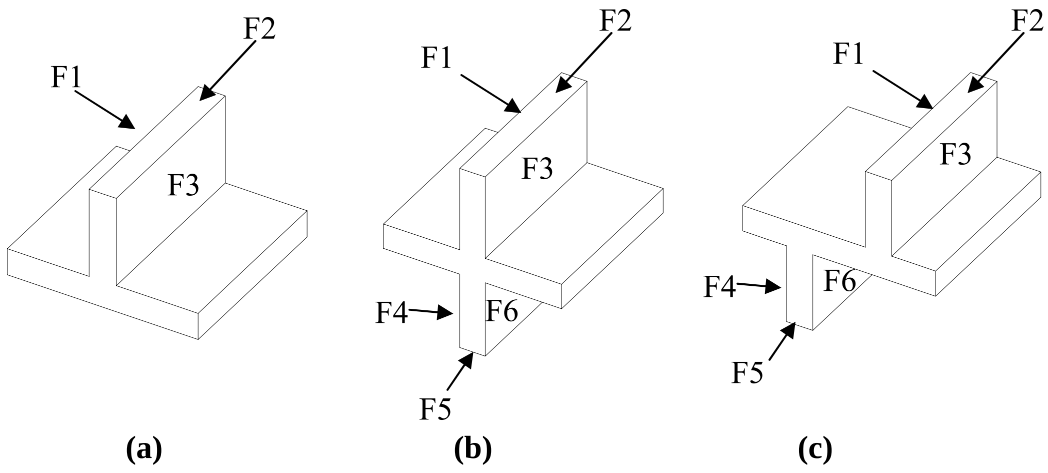


Figure 3.1 Feature Recognition of Slot Features

The types of junction features shown in Figure 3.1 are very common on moulded parts, and it is essential that a feature recognition approach for these parts is able to identify and differentiate between the various wall junction types. The number and types of wall intersections on a typical moulded part mean that it is not feasible to use existing graph based techniques to perform the feature recognition and an alternative approach is required.

3.3 Mid-Surface Representation for Moulded Parts

As already described in the literature review the mid-surface of a part is a dimensionally reduced representation in which each wall is represented by a surface with zero thickness. The mid-surface can be visualised as the locus of the centre of a maximal sphere rolling around the interior of the part.

For thin walled parts the mid-surface abstraction offers a simplified representation that retains the main design characteristics of the original part. Mid-surface representations have been widely used in engineering analysis for thin walled moulded parts. For example in finite element analysis it is common to use a mid-surface abstraction of a thin walled part to reduce the computational cost of performing the finite element analysis and several flow analysis simulation tools for injection moulding use a mid-surface abstraction for mould filling simulations. A mid-surface model is suitable to represent the shape of cast and injection moulded parts because these manufacturing processes require that parts be designed with thin and relatively constant wall thickness.

3.3.1 Mid-surface Model Evaluation for Moulded Parts

There are several algorithms that can generate a mid-surface abstraction from a CAD solid model in an automatic or semi-automatic process (these have previously been described in Section 2.3). The algorithms all generate a geometric representation of the mid-surface geometry and an associated numerical value or function to represent the thickness of each mid-surface face. The problem with mid-surface generation algorithms is that it is not possible to guarantee that a representative mid-surface can be generated for any arbitrary solid part. In particular it may not be possible to compute a representative mid-surface for parts that have thick sections or “chunky” regions.

An evaluation of one commercially available mid-surface algorithm has been performed as part of this research to determine whether the mid-surface representation can be used reliably as a basis for feature recognition from moulded parts. The results of this evaluation have been published by the author (Lockett and Guenov, 2002).

The mid-surface algorithm that has been evaluated is the EDS I-DEAS NX mid-surfacing function. This function was selected because it is a relatively mature commercial implementation that was available to the author, however mid-surfacing tools are also available in other CAE systems (for example Pro/ Engineer, MSC/ PATRAN) and in dedicated applications (for example the medial-object from TranscenData). Mid-surface algorithms use a variety of techniques to compute the mid-surface of a solid object. The I-DEAS function automatically identifies candidate pairs of faces from the solid part and generates a mid-surface between each face pair; it then uses surface extension and trim operations to connect the faces to create the overall mid-surface representation of the complete part.

The ability of mid-surface algorithms to produce a good quality mid-surface abstraction is dependent of the shape characteristics of the geometry that is presented to them. Table 3.1 shows an example of the mid-surface generation for a simple injection moulding test part based on an example shown in McMahon and Browne (1998). Five variants of the part have been modelled by the author using the I-DEAS software, with the main wall thickness (t) varying from 1.5 mm to 7.5 mm and protrusions from the main wall modelled at 66% of the main wall thickness. A mid-surface abstraction has been generated for each variant of the model using the I-DEAS mid-surface function, and the results for each model are shown in the figure.

The results show that when the wall thickness is small relative to the other dimensions of the part (in this case for wall thicknesses less than or equal to 4.5 mm) the mid-surface accurately represents all the features of the part. The boss, rib, and buttress features are all clearly visible in the mid-surface model, and the number of faces in the model has been significantly reduced from 38 to 21. When the wall thickness is increased to above 4.5 mm the software becomes less able to generate representative mid-surfaces. In this example when t = 6 mm the boss features become too small relative to the wall thickness to appear as faces on the mid-surface, so the bosses appear only as holes in main wall. When t = 7.5 mm the overall shape of the part becomes distorted because the fillet and buttress features cannot be resolved. In this example the mid-surface representation is not valid when the wall thickness is greater than or equal to 6 mm, however if this part were to be manufactured using injection-moulding a main wall thickness of 2 – 4 mm would be appropriate, and wall thicknesses of greater than 4 mm are likely to cause cooling problems, and be uneconomic to mould. The results are therefore acceptable for this part.

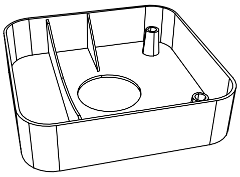
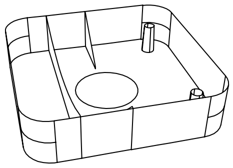
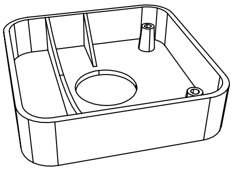
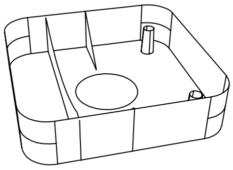
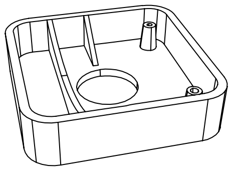
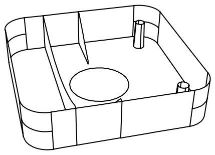
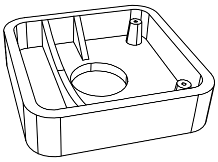
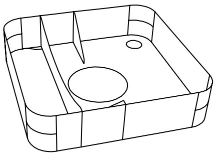
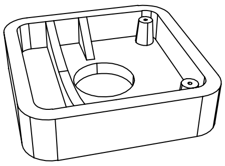
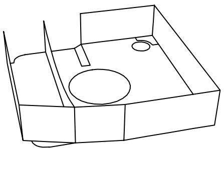
Wall Thickness (t)	Solid Part	Mid-surface Abstraction
1.5 mm		
3.0 mm		
4.5 mm		
6.0 mm		
7.5 mm		

Table 3.1 Example Of Mid-Surface Generation for a Simple Moulded Part (Lockett & Guenov, 2002)

The I-DEAS mid-surface function has been tested for a range of moulded parts to demonstrate that the tool can be used to generate mid-surfaces for a representative range of designs. Figure 3.2 shows some examples of representative moulded parts and mid-surfaces generated using I-DEAS.

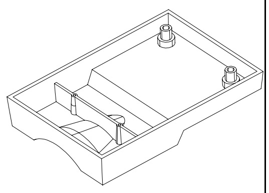
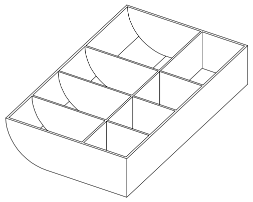
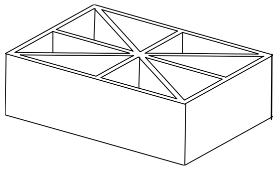
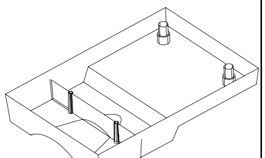
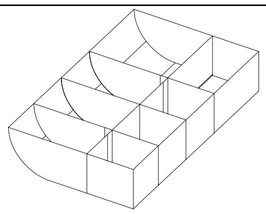
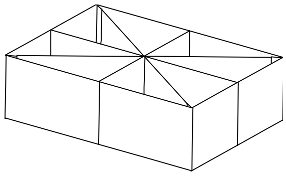
	Cover Part	**Coin Tray**	**Eight part box**
Solid Model			
Mid-surface Abstraction			

Figure 3.2 Example of Mid-Surface Generation for Moulded Parts

The evaluation of the I-DEAS mid-surface function has shown that the software is able to generate mid-surfaces for a range of representative moulded parts. However the function has also been shown to have some limitations. The conclusions of the capability of the I-DEAS mid-surface function are summarised below:

1. The function gives best results for parts that have a thin and relatively uniform wall thickness. If a part has chunky regions a mid-surface abstraction may be generated that does not provide a meaningful representation of the original shape. The most extreme example of this problem is that the mid-surface representation of a sphere would be a point at the sphere’s centre.
2. Features that are small relative to the wall thickness of the part may not appear on the mid-surface
3. In some cases the surface trimming and extension operations do not correctly operate to form a fully connected part.
4. Occasionally it is necessary to make some manual interventions in the mid-surface process (for example to assist with the selection of face pairs)

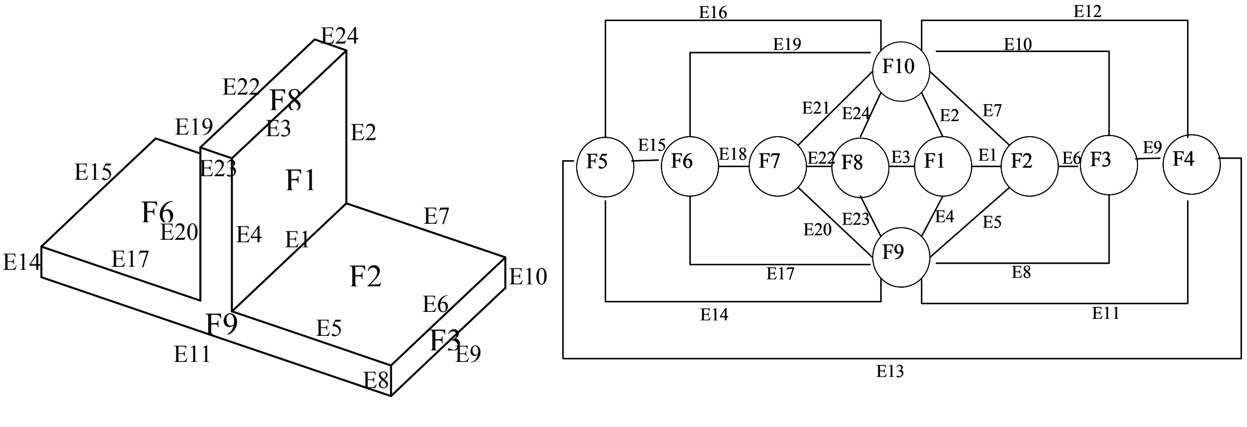
Although these limitations mean that it is not always possible for automatically generate the abstraction required for feature recognition, it is believed that as mid-surface generation techniques continue to be an active area of research the mid-surfacing algorithms will improve in the future. The results were sufficiently encouraging to pursue the mid-surface as the basis of a feature recognition methodology for moulded parts. In Chapter 4 a technique will be presented to evaluate the quality of the mid-surface that is generated, to give confidence in the results that are obtained.

3.4 Mid-surface Model Graph Structure

A feature recognition methodology for thin-walled parts has been developed in this research that uses the mid-surface abstraction as the basis for feature recognition. The feature recognition methodology uses a graph based approach, but is significantly different from other graph based approaches because the non-manifold mid-surface geometry cannot be represented using the standard geometry graphs that have previously been used for feature recognition.

Most graph based feature recognition techniques construct a separate data structure to represent the geometry and facilitate the search for features. Features are recognised by parsing a graph structure constructed from the model or matching sub-graphs to predefined templates. For example Shah, and Mantyla (1995) present a commonly used approach which uses a face adjacency graph (FAG) in which the nodes of the graph represent the faces of the object, and the arcs represent the connectivity (edges) between those faces.

An example is shown in Figure 3.3 in which a simple T-shaped part has been modelled as a B-rep solid model. The model has 10 faces and 24 edges and the associated FAG graph has ten nodes representing the faces of the part, and 24 arcs representing the edges connecting the faces. The feature recognition techniques that use this type of geometry graph classify edges as concave, convex or smooth, and then search for patterns of concave/ convex edges in the graph to recognise features (Joshi and Chang, 1988). In the example in Figure 3.3 the sequence of faces F2, F1, F8, F7, F6 represents a rib feature, and the rib is characterised geometrically by the sequence of concave, convex, convex, concave edges between the five faces. The FAG can be represented as a 10 x 10 matrix in which the rows and columns represent the faces of the part. A "1" in a matrix cell indicates the existence of an edge connecting two faces, and a "0" indicates that two faces are not connected.



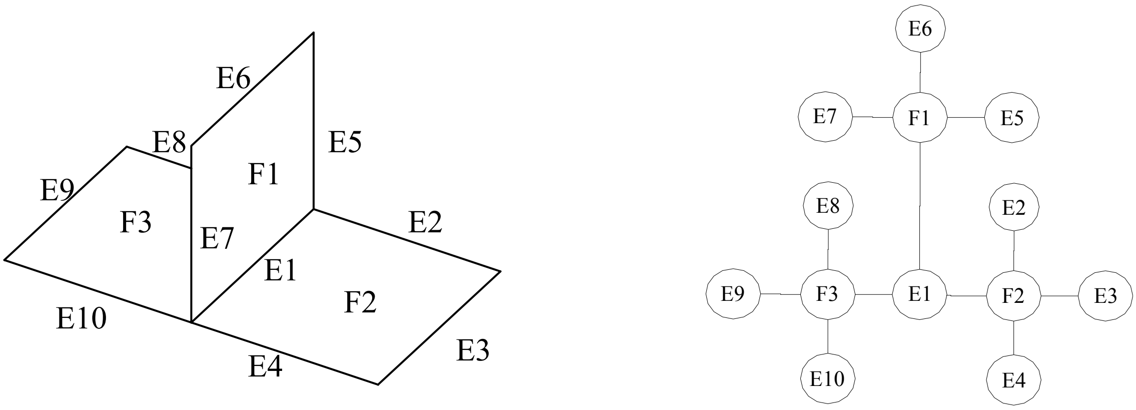
$$\begin{array}{c c}
 & \begin{matrix} F1 & F2 & F3 & F4 & F5 & F6 & F7 & F8 & F9 & F10 \end{matrix} \\
\begin{matrix} F1 \\ F2 \\ F3 \\ F4 \\ F5 \\ F6 \\ F7 \\ F8 \\ F9 \\ F10 \end{matrix} &
\begin{bmatrix}
0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\
1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\
0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\
1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0
\end{bmatrix}
\end{array}$$

Figure 3.3 Face Adjacency Graph and Matrix for Simple T- Shaped Part.

The FAG is suitable for representing the topology of manifold solid models in which every edge must connect exactly two faces. However, this representation is not suitable for representing non-manifold geometries because in a non-manifold geometry an edge may connect any number of faces

An alternative graph structure has been proposed in this research to represent the mid-surface model topology for feature recognition. The Mid-surface Adjacency Graph (MAG) is able to represent mid-surface models in which the faces are connected along shared edges in any combination and do not conform to the requirements of a manifold solid model. The MAG represents both the faces and edges of the model as nodes in the graph, and the connectivity between the faces and edges is represented by the arcs of the graph.

Figure 3.4 (a) shows the mid-surface representation for the T-shaped part. The mid-surface model contains three faces representing the three walls of the part, and 10 edges bounding those faces. The MAG for the T-shaped part is shown in Figure 3.4 (b) and has 13 nodes representing the geometry entities in the model (three faces, and 10 edges), and 12 arcs representing the connectivity between the entities. By observation from the MAG shown in Figure 3.4 (b) it is clear that edge *E1* represents a junction between the three faces because it is connected to faces *F1*, *F2,* and *F3*. This property of the MAG provides much more direct access to wall junctions than is available from the original B-rep CAD geometry and FAG.



(a) **(b)**

Figure 3.4 Mid-Surface Model & Adjacency Graph for T-Junction Part

The MAG describes the face-edge adjacency for the mid-surface model, but there is other important topological information that is required for feature recognition, and the MAG definition has therefore been augmented to capture additional topological attributes. The attributed MAG (AMAG) defines three attributes associated with each face – edge connection:

i. Edge Loop Number - specifies which edge loop contains the edge on the face.
ii. Internal/ External – specifies whether the edge belongs to an internal or external face bound
iii. Number of Uses – specifies the number of times the edge is used in the face

The AMAG represents these attributes as indices on the graph arcs. For clarity only the second and third attributes are shown in the figures, but the first is also important during feature recognition. Figure 3.5 shows the AMAG for the T-shaped part showing the two attributes for each arc. The first attribute specifies whether the edge belongs to an internal or external edge (n=1 for external, n>1 for internal)[1] and the second index specifies the number of times the edge is used in the face. For this very simple example all the edges belong to the external loop, and each edge is used only once in each face.

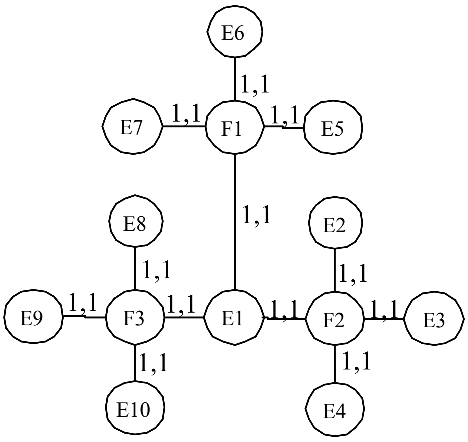


Figure 3.5 Attributed Mid-surface Adjacency Graph for T-Shaped Part

3.4.1 Face-Edge Adjacency Matrix

The AMAG can be represented as a face-edge adjacency matrix which captures the adjacency between faces and edges in the graph. The complete face-edge adjacency

[1] Note that the numbering scheme for internal/ external edges has been changed since the publication (Lockett & Guenov, 2005) to match the implementation of the feature recognition software.

matrix for a graph with m edges and n faces is an $(n+m) \times (n+m)$ *matrix*, but since only the face to edge connectivity is required only an $n \times m$ a subset of the matrix is required to capture the graph topology.

The face-edge adjacency matrix for the T-shaped part is the 3 x 10 matrix shown in Figure 3.6. The values in the matrix m represent the connectivity between the faces and edges in the graph, where $m_{ij} = 1$ means that edge j is a bounding edge of face i, and $m_{ij} = 0$ means that edge j is not a bounding edge of face i.

$$\begin{array}{c|cccccccccc|}
 & E1 & E2 & E3 & E4 & E5 & E6 & E7 & E8 & E9 & E10 \\
F1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\
F2 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
F3 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1
\end{array}$$

Figure 3.6 Face-Edge Adjacency Matrix for T-Shaped Part

The three face-edge attributes that have been defined to allow the AMAG to capture additional topological relationships can be represented by constructing a three dimensional matrix of edges, faces and attributes. The attributed face-edge adjacency matrix for the T-shaped part is therefore a 3 x 10 x 3 matrix. Once the adjacency matrix has been constructed it can be used to provide some important properties of the model to aid feature recognition. For example the number of non-zero values in each edge column provides the number of faces which are connected along an edge, and the number of edge loops in a face row defines the number of inner loops (face features) in the face.

3.4.2 Mid-surface Feature Classification

Before describing the feature recognition approach in detail it is important to define the types of moulding features that will be recognised. Injection-moulding and casting processes are able to manufacture parts with a wide range of curved and styled shapes, but in general even though the external shape of the part may be styled the attached features are usually functional features that can be classified using a relatively small number of types. The aim of the feature recognition process is to identify design features that will have an impact on the manufacturability of the part. Some common features types that are important for moulded parts are:

- Ribs and buttresses – used to provide strength or stiffness to the part
- Bosses – used as mounting and reinforcing points
- Holes – to allow access or fastening
- Wall Junctions – may affect mould filling or mould cooling behaviour

The design of these features can have a major impact on the manufacturability, cost and quality of moulded parts. The moulding features have been classified into three main types:- face features, junction features and stiffener features. These feature classes are described in more detail below, and are shown in Table 3.2.

Face features are features that are connected to the interior of a face, and are not connected to any other faces in the part. A face feature may be composed of zero, one or more faces. Three face features types have been defined for the feature library – hole, boss and fin.

Junction features are features that represent a junction between two or more faces. Junction features are used as building blocks for stiffener features, as well as defining relationships between groups of stiffener features. The subtype of a junction feature is defined by the number of faces meeting along a common edge, and by the angles between the connected faces.

Stiffener features are features that provide additional strength or stiffness to a part. Stiffeners are characterised by faces that are connected to two or more adjacent faces of the part main wall. Two types of stiffener feature have been defined – rib and buttress.

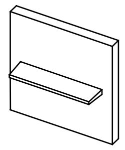
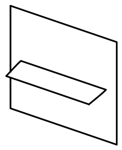
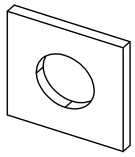
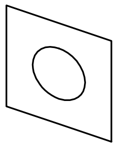
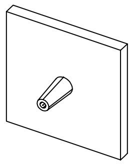
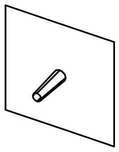
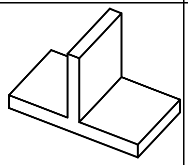
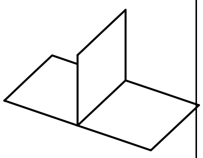
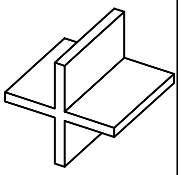
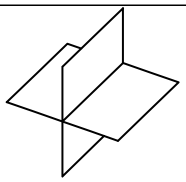
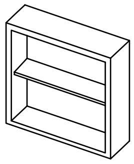
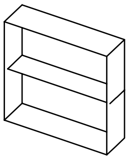
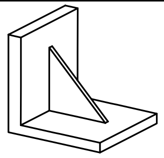
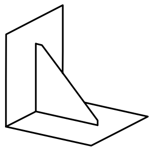
Feature Class	Feature Type	Solid Model	Mid-surface Model	Manufacturability Impact
Face Feature	**Fin**			Fins may affect the external wall quality at the attachment. Careful design of fin proportions can minimise problems (Boothroyd, Dewhurst and Knight, 2002)
	Hole			Small holes may be more economic to drill,may cause weld lines (Boothroyd, Dewhurst and Knight, 2002)
	Boss			The proportions of bosses are important for main wall quality. Supporting ribs may be required to react lateral forces (Ticona, 2000)
Junction Features	**T-Junction**			High order junctions may cause sink marks or warping. Wall thickness proportions and angles are important (Boothroyd, Dewhurst and Knight, 2002)
	X-Junction			High order junctions may cause sink marks and warping. Where possible should be redesigned as two staggered junctions(Boothroyd, Dewhurst and Knight, 2002)
Stiffener Features	**Rib**			Ribs may cause warping or appearance problems – rib proportions are important (Boothroyd, Dewhurst and Knight, 2002)
	Buttress			Buttresses may cause warping or appearance problems – proportions are important (Boothroyd, Dewhurst and Knight, 2002)

Table 3.2 Summary of Feature Types and Graph Characteristics

(Lockett & Guenov, 2005)

The initial library that has been defined contains a relatively small set of moulding features, which have been selected to be representative of common functional features that are used to design moulded parts. Additional types of features could be added to the library within the current classifications, for example a bridge feature that connects two face features might be considered, or a composite feature that is composed of several smoothly connected adjacent faces.

3.5 Mid-surface Feature Recognition

The feature recognition methodology that has been developed in this research uses an algorithmic approach to recognise features from the attributed mid-surface adjacency graph (AMAG). The feature recognition process is undertaken in three stages:- firstly the mid-surface model is parsed to construct the AMAG and the graph is stored as an attributed face-edge adjacency matrix, then feature recognition algorithms perform an initial feature identification using the adjacency matrix and finally the feature recognition is completed by performing geometric evaluations on the identified features.

Table 3.3 shows the mid-surface geometry, adjacency graph and graph characteristics for the library of feature types defined in Table 3.2. The graph segments are shown with two attributes associated with each graph arc; the first attribute identifies whether the edge is connected to an internal or external edge loop within the face (n=1 for external and n>1 for internal), and the second attribute identifies how many times the edge is used in the face. For clarity the edge loop identifiers are not shown in this figure.

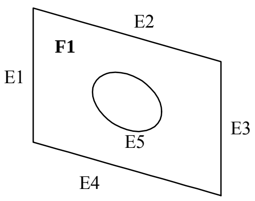
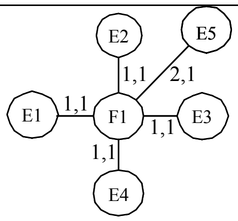
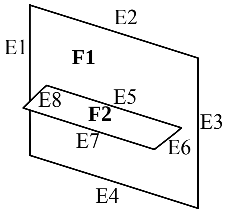
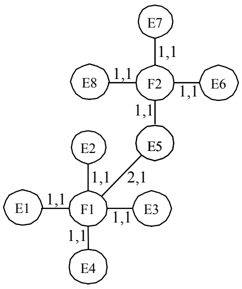
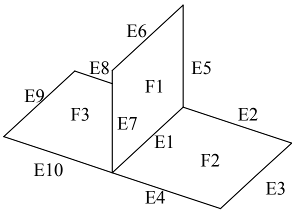
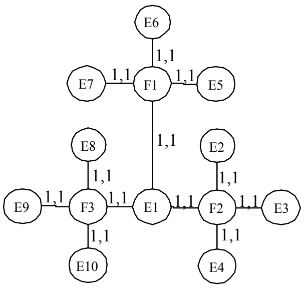
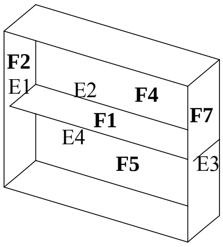
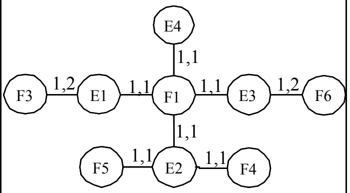
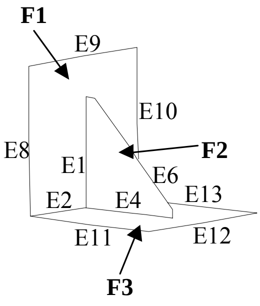
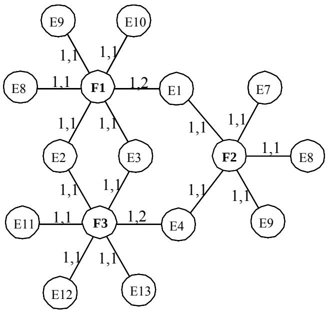
Feature Class	Feature Type	Mid-surface geometry	Attributed Mid-surface Adjacency Graph	Graph Characteristics
Face Feature	**Hole**		 E2 E5 1,1 2,1 E1 1,1 F1 1,1 E3 1,1 E4	A **hole feature** is recognised by the existence of an internal edge loop that is not connected to any other faces (e.g E5).
	Fin		 E7 1,1 E8 1,1 F2 1,1 E6 1,1 E2 E5 1,1 2,1 E1 1,1 F1 1,1 E3 1,1 E4	A **fin feature** is recognised by the existance of an internal edge loop that is connected to another face along a single edge. A boss may be topologically identical to a fin, but is distinguished by its geometric characteristics
Junction Features	**T-Junction**	 E6 E8 E5 E9 F1 F3 E7 E2 E1 F2 E10 E4 E3	 E6 1,1 E7 1,1 F1 1,1 E5 1,1 E8 E2 1,1 1,1 E9 1,1 F3 1,1 E1 1,1 F2 1,1 E3 1,1 1,1 E10 E4	A **junction feature** is recognised as an edge that is connected to more than one face (e.g. E1). An attribute representing the angles between the faces is also required to complete the feature recognition. The order of the junction is defined as the number of faces using the edge.
Stiffener Features	**Rib**		 E4 1,1 F3 1,2 E1 1,1 F1 1,1 E3 1,2 F6 1,1 F5 1,1 E2 1,1 F4	A **rib feature** is recognised as a face with several adjacent high order edges and at least one unconnected edge (e.g. F1). If all the edges are connected with high order junctions the face is not a rib but is a surrounded face.
	Buttress	 F1 E9 E10 E8 F2 E1 E6 E2 E4 E13 E11 E12 F3	 E9 E10 1,1 1,1 E8 1,1 F1 1,2 E1 E7 1,1 1,1 1,1 1,1 E2 E3 F2 1,1 E8 1,1 1,1 1,1 1,1 E11 1,1 F3 1,2 E4 E9 1,1 1,1 E12 E13	A **buttress feature** is recognised as a face with two adjacent edges connected at order 3 or higher, with the remaining edges unconnected. Note that high order junctions may be formed from reused edges that extend into the interior of a face(e.g. E1 and E4)

Table 3.3 AMAGs for Simple Feature Types (Lockett & Guenov, 2005)

The detailed feature recognition algorithms for the main feature types are described in more detail in the following sections.

3.5.1 Face Features

Face features are classified as features that are attached to the interior of a face and may be holes, slots, fins, bosses or other attached features. The graph segments for simple hole and fin features can be seen in Table 3.3.

The first step towards recognising a face feature is to use the attributed adjacency matrix to find faces that have internal edge loops. Once an internal edge loop has been identified the face feature recognition algorithm checks which (if any) faces are connected to each of the edges in edge loop. If attached faces are found for the edge loop then further geometric checks are performed to determine the type of attached face feature, otherwise if no attached faces are found then the edge loop is identified as a hole. For example a boss is recognised by a cylindrical or conical face that is connected to the parent face by a circular edge and a fin is recognised as a planar face that is connected to its parent face along a single edge. The algorithm for face feature recognition takes as its input an array of faces with internal edge loops, and an array of edge orders, both of which can be obtained from the adjacency matrix. The face feature recognition is performed at the beginning of the feature recognition process so that face features can be disregarded for the remainder of the feature recognition process. The face-feature recognition algorithm is shown below:

```
    Algorithm   Find_Face_Features
    begin
            Let FACES-WITH-INTERNAL-LOOPS be a list of all the
            faces in the model with internal edge loops
            for each face f in FACES-WITH-INTERNAL-LOOPS do
                  for each internal loop l in f do
                        ATTACHED-FACES[l] = {}
                        for each edge e in l do
                              if EDGE-ORDERS[e] > 1 then
                                          Append adjacent faces
                                          to ATTACHED-FACES[l]
                        end
                  end
                  if ATTACHED-FACES[l] = {} then
                        Edge loop l is a hole
                              else
                        Edge loop l is a face feature with
                        attached faces ATTACHED-FACES[l]
            end
end
```

3.5.2 Junction Features

Junction features are building blocks towards stiffener features and can be identified using the mid-surface adjacency matrix. The graph segment for a sample junction feature is shown in Table 3.3. The order of a junction is defined by the number of faces connected along a shared edge, and an edge is referred to as a high-order edge if it connects at least three faces.

The order of an edge can be obtained by counting the number of non-zero elements in the relevant column of the adjacency matrix. After initial identification, the junction feature recognition is completed by calculating the angles between the faces connected along the edge. For example a 2nd order junction may be classified as a V, or L junction depending on the face angles.

3.5.3 Stiffener features

Stiffener features are structural elements that are used to add strength or stiffness to the part. The graph segments for simple stiffener features can be seen in Table 3.3. The stiffener feature recognition algorithm identifies stiffener features by searching for bounding edge loops that have specific patterns of adjacent high order and unconnected edges.

The first step towards recognising a stiffener feature is to identify a face with at least two adjacent high order edges, and at least one unconnected edge. The type of the stiffener feature is then refined by counting the number of adjacent high order and unconnected edges. A rib is defined as a stiffener with at least three adjacent high order edges and at least one unconnected edge, and a buttress is defined as a stiffener with at least two adjacent high order edges and at least one unconnected edge. A face with all its bounding edges connected to other faces is defined as a surrounded face.

The inputs to the stiffener feature recognition algorithm are an ordered list of edges for each bounding edge loop, and array of edge-orders for each edge in the model. The algorithm for stiffener feature recognition is shown below:

```
Algorithm Find_Stiffeners
    begin
         for each face f do
          let oel be the identifier for the outer edge loop
          let LIST-OF-EDGES[oel] be an ordered list of the
          edges in oel
          NO-OF-ADJ-HO-EDGES = 0 ; adjacent high-order edges
          MAX-ADJ-HO-EDGES = 0; max no. of adjacent high-order
          edges in loop
          NO-OF-UNCON-EDGES = 0 ; unconnected edges
          for each edge e in LIST-OF-EDGES[oel] do
               if EDGE-ORDERS[e] >= 3 then
                NO-OF-ADJ-HO-EDGES = NO-OF-ADJ-HO-EDGES + 1
                if NO-OF-ADJ-HO-EDGES > MAX-ADJ-HO-EDGES  then
                      MAX-ADJ-HO-EDGES = NO-OF-ADJ-HO-EDGES
                end
               Else if EDGE-ORDERS[e] = 2 then
                NO-OF-ADJ-HO-EDGES = 0
               else if EDGE-ORDERS[e] =1 then
                NO-OF-UNCON-EDGES = NO-OF-UNCON-EDGES+ 1
                NO-OF-ADJ-HO-EDGES = 0
                end
          end
          if 2 <= MAX-ADJ-HO-EDGES < NO-OF-EDGES
          and NO-OF-UNCON-EDGES >=1
          then face f is a stiffener feature
          end
    end
```

3.6 Feature Recognition Algorithm Complexity

The algorithm complexity of the mid-surface feature recognition approach presented in this chapter has been investigated for comparison with other graph based approaches. The worst case O(g) complexity method has been used for the comparison because it is a measure that is commonly used in the literature, although it has been found to be difficult to apply sensibly for feature recognition applications.

The worst case complexity measure states that the complexity for a sequence of operations is dominated by the most expensive operation (rule of sums). This means that if the various algorithms are performed in series then only the most expensive algorithm needs to be considered. The most expensive algorithm in the mid-surface feature recognition approach is the find_face_features algorithm which has 4 nested loops.

The algorithm complexity for the find_face_features algorithm can be stated as $O(F*L_F*E_L*F_E)$ where F is the number of faces in the part, L_F is the maximum number of loops per face, E_L is the maximum number of edges per loop, and F_E is the maximum number of faces per edge; which gives a worst case complexity of $O(n^4)$. However, in reality this gives an overly high estimate because only F increases directly with the problem size. The algorithmic complexity is therefore somewhere between O(n) and $O(n^4)$, but it is difficult to define a more precise value.

The algorithm complexity of other feature recognition techniques has been reviewed using data from the literature for comparison with the mid-surface approach. Graph based feature recognition is based on sub-graph matching (isomorphism), and is known to be a highly computationally intensive technique. Shah and Mantyla (1995) state that "Graph-based methods …must perform exhaustive searches of feature patterns in a (potentially large) boundary representation data structure". They state that the worst case algorithm complexity for graph matching is O(n!) which is worse than exponential.

Han, Pratt and Regli (2000) argue that for practical implementations of graph based feature recognition worst-case complexity analysis is not appropriate, because the sub-graphs to be matched are always of small size even if the part graph is large. They state that "algorithms for computing sub-graph isomorphism are of polynomial time complexity when the size of the graph to be matched is a constant". They calculate the complexity for finding all the instances of a sub-graph of size *k* in a part graph with *n* nodes as $O(n^k)$.

Gao and Shah (1998) calculate the computational complexity of their hybrid approach for machining feature recognition to be O(M*N*T*(M+N)), where M is the maximum number of edges per face, T is the maximum number of arcs in the graph and N is the maximum number of nodes in the graph. Using the standard definition of worst case complexity this could be expressed as $O(n^4)$, but using the same argument as Han, Pratt and Regli this value could be seen as an overly high estimate. The value of M is not directly related to the problem size, so the complexity could be stated as between $O(n^3)$ and $O(n^4)$.

The algorithm complexity evaluation for the mid-surface feature recognition approach has calculated the worst case complexity to be less than $O(n^4)$, which compares favourably with other graph based feature recognition techniques. This result is in line with expectations because the mid-surface approach does not require the expensive searching for feature interactions that are required for other feature recognition techniques. However, the worst case algorithmic complexity does not take into account other such as the cost of generating of the mid-surface abstraction, that also contribute to the cost of the mid-surface approach.

3.7 Feature Recognition Example

A simple feature recognition example is presented in this section to illustrate how the methodology can be applied to a sample part. The test case that has been used is deliberately very simple so that it is possible to follow the feature recognition manually.

The geometry that has been used for the test case is shown in Figure 3.7 and is a simple two part box part with two holes.

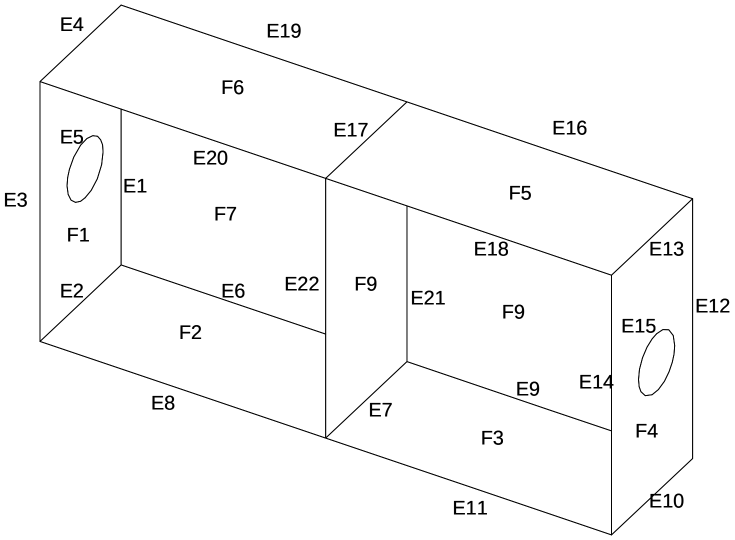
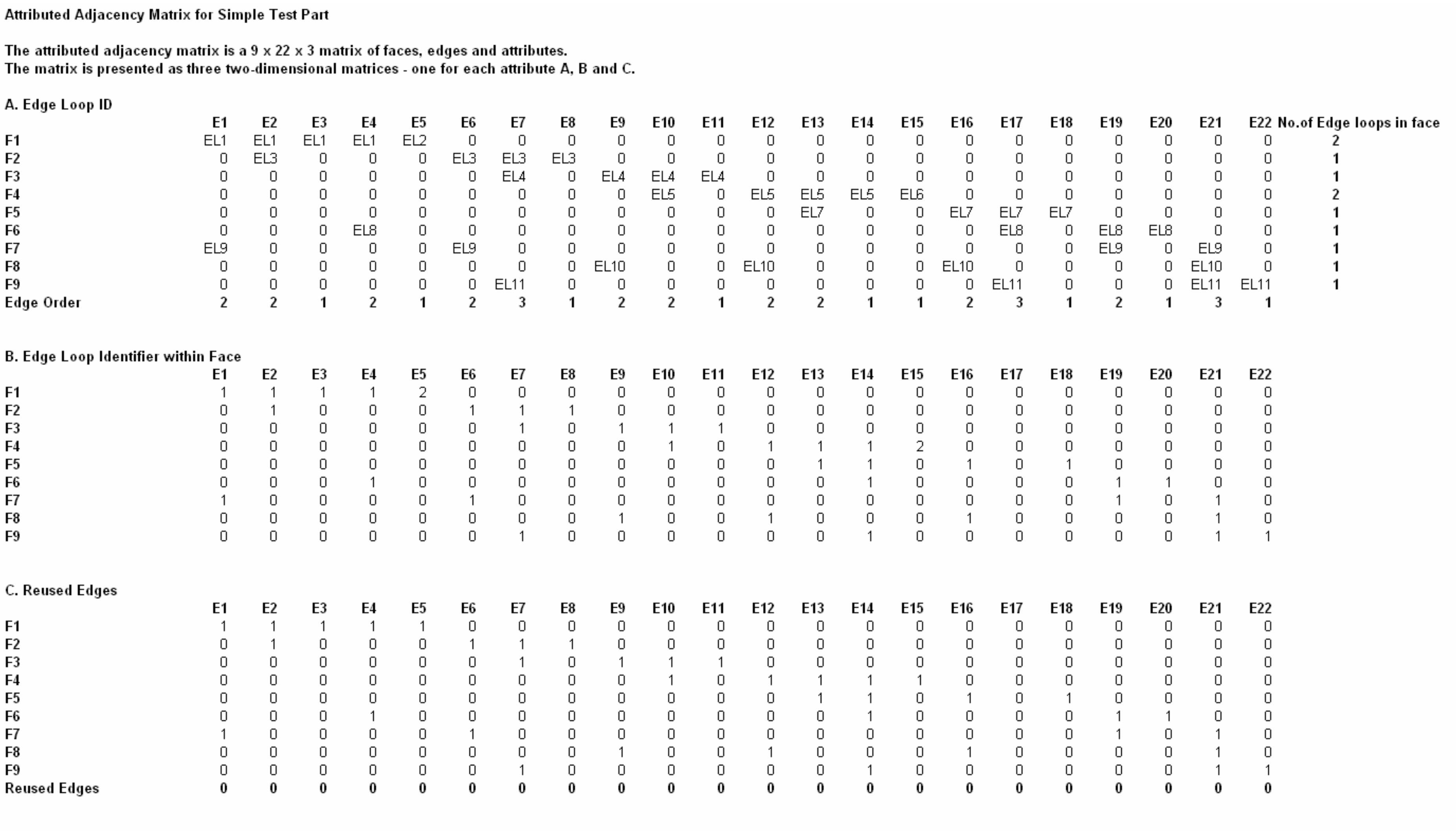


Figure 3.7 Mid-surface model of Simple Test Part

The first step in the feature recognition process is to construct the mid-surface adjacency graph for the part. The attributed adjacency matrix for the graph is included in Figure 3.8 and a graphical representation of the geometry graph is shown in Figure 3.9.



Attributed Adjacency Matrix for Simple Test Part

The attributed adjacency matrix is a 9 x 22 x 3 matrix of faces, edges and attributes.
The matrix is presented as three two-dimensional matrices - one for each attribute A, B and C.

A. Edge Loop ID

	E1	E2	E3	E4	E5	E6	E7	E8	E9	E10	E11	E12	E13	E14	E15	E16	E17	E18	E19	E20	E21	E22	No.of Edge loops in face
F1	EL1	EL1	EL1	EL1	EL2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2
F2	0	EL3	0	0	0	EL3	EL3	EL3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
F3	0	0	0	0	0	0	EL4	0	EL4	EL4	EL4	0	0	0	0	0	0	0	0	0	0	0	1
F4	0	0	0	0	0	0	0	0	0	EL5	0	EL5	EL5	EL5	EL6	0	0	0	0	0	0	0	2
F5	0	0	0	0	0	0	0	0	0	0	0	0	EL7	0	0	EL7	EL7	EL7	0	0	0	0	1
F6	0	0	0	EL8	0	0	0	0	0	0	0	0	0	0	0	0	EL8	0	EL8	EL8	0	0	1
F7	EL9	0	0	0	0	EL9	0	0	0	0	0	0	0	0	0	0	0	0	EL9	0	EL9	0	1
F8	0	0	0	0	0	0	0	0	EL10	0	0	EL10	0	0	0	EL10	0	0	0	0	EL10	0	1
F9	0	0	0	0	0	0	EL11	0	0	0	0	0	0	0	0	0	EL11	0	0	0	EL11	EL11	1
Edge Order	**2**	**2**	**1**	**2**	**1**	**2**	**3**	**1**	**2**	**2**	**1**	**2**	**2**	**1**	**1**	**2**	**3**	**1**	**2**	**1**	**3**	**1**	

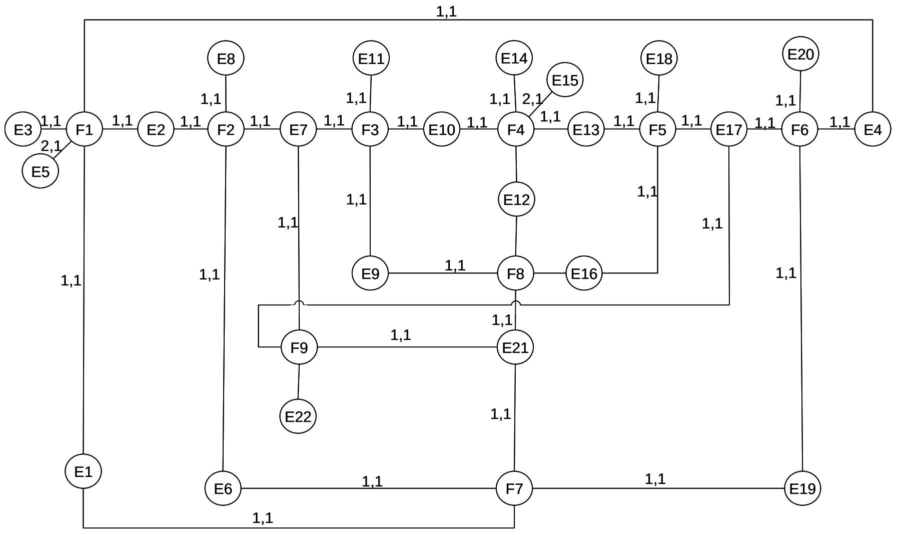
B. Edge Loop Identifier within Face

	E1	E2	E3	E4	E5	E6	E7	E8	E9	E10	E11	E12	E13	E14	E15	E16	E17	E18	E19	E20	E21	E22
F1	1	1	1	1	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
F2	0	1	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
F3	0	0	0	0	0	0	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0
F4	0	0	0	0	0	0	0	0	0	1	0	1	1	1	2	0	0	0	0	0	0	0
F5	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	0	1	0	0	0	0
F6	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	1	0	0
F7	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0
F8	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	1	0	0	0	0	1	0
F9	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	1	1

C. Reused Edges

	E1	E2	E3	E4	E5	E6	E7	E8	E9	E10	E11	E12	E13	E14	E15	E16	E17	E18	E19	E20	E21	E22
F1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
F2	0	1	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
F3	0	0	0	0	0	0	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0
F4	0	0	0	0	0	0	0	0	0	1	0	1	1	1	1	0	0	0	0	0	0	0
F5	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	0	1	0	0	0	0
F6	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	1	0	0
F7	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0
F8	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	1	0	0	0	0	1	0
F9	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	1	1
Reused Edges	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

Figure 3.8 Attributed Adjacency Matrix for Simple Test Part.



Edge Loops:

EL1 = E1, E2, E3, E4

EL2 = E5

EL3 =E2, E6, E7, E8

EL4 = E7, E9, E10, E11

EL5 = E10, E12, E13, E14

EL6 = E15

EL7 = E13, E16, E18, E17

EL8 = E17, E19, E4, E20

EL9 = E1, E6, E21, E19

EL10 = E9, E12, E16, E21

EL11 = E7, E22, E17, E21

Figure 3.9 Mid-surface Adjacency Graph for simple test part

The second step is to identify the face features in the part. Face features are attached to internal edge loops in the model, so it is necessary to identify all the faces in the model that have internal edge loops. The find_face_features algorithm is applied to all the faces with internal loops to identify the types of features on each face.

In this example it can be seen from the adjacency matrix (B) that there are two internal edges (edge E5 on face F1, and edge E15 on face F4). The identifiers for the two internal edge loops containing these edges can be obtained from the same array elements in adjacency matrix (A) (edge loops EL2 and EL6 respectively). Finally the edge orders for each edge in the two edge-loops can be obtained from the bottom row of the adjacency matrix (A) (the orders of E5 and E15 are both 1 indicating that they are not connected to any other faces). The face feature recognition for this part finds two internal edge loops EL2 and EL6 which each contain one unconnected edge. The two face features are therefore recognised as holes.

The third step in the feature recognition process is to identify the stiffener features. Stiffener features are identified by faces with external edge loops that have particular patterns of high order and unconnected edges. The find_stiffeners algorithm is applied to each external edge loop in the model. The algorithm cycles through the edges in each edge loop and checks the order of each edge. In this example edge loop EL11 on face F9 contains four edges {E7, E22, E17, E21} and the orders of those edges are {3, 1, 3, 3} which means that the face is connected has high order connections on three adjacent edges, and has one unconnected edge. Face F9 is therefore recognised as a rib. Further geometric checks should then be performed to validate the features that have been recognised.

3.8 Summary and Conclusions

In this chapter a novel feature recognition approach for moulded parts has been presented. It has been shown that feature recognition for moulded parts requires a different approach to existing machining feature recognition techniques, and a feature recognition methodology that uses a mid-surface abstraction from the solid geometry has been proposed. A library of common moulding features has been defined, which could be further extended to be applicable to a wider range of parts.

The mid-surface based feature recognition uses a non-manifold mid-surface abstraction from the part geometry to construct an attributed mid-surface geometry graph as the basis for feature recognition. The feature recognition then uses an algorithmic approach to identify candidate moulding features, and completes the feature recognition by performing geometric checks.

The advantage of the mid-surface based approach is that important topological relationships are more directly accessible than from the complete solid geometry, but the disadvantage is that the feature recognition is performed on an abstraction from the true geometry and the quality of the feature recognition results is dependent on the quality of the mid-surface abstraction that is generated. The following chapter will describe techniques that have been developed to evaluate the mid-surface quality and feature recognition results.

4 EVALUATION TECHNIQUES

4.1 Introduction

The preceding chapter has described a novel feature recognition methodology for moulded parts based on a mid-surface abstraction. Mid-surface feature recognition has been shown to have some benefits over existing feature recognition techniques for moulded parts, but it also introduces new problems because the feature recognition is performed on a geometry abstraction and not the actual part geometry.

This chapter describes the techniques that have been developed in this project to evaluate the quality of the feature recognition. Two metrics that have been developed to evaluate the feature recognition results: firstly a measure of the mid-surface quality, and second a measure of the feature recognition quality.

4.2 Mid-surface Quality Evaluation

A mid-surface model is a dimensionally reduced representation in which each wall is represented by a surface with zero thickness. Existing mid-surface generation techniques have already been described in the Literature Review (Section 2.3), and a commercial mid-surface generation tool has been evaluated in Section 3.3. The review of existing mid-surface tools has identified that although there are several commercial implementations of mid-surface generation software there is no algorithm that can automatically construct the mid-surface for any arbitrary geometry.

The limitations that have been identified for mid-surface generation tools mean that if a functional feature recognition tool is to be developed using a mid-surface approach then it will be important to be able to evaluate the accuracy of the mid-surface geometry before proceeding with the feature recognition process. Two approaches to evaluating the mid-surface quality have been investigated: a volumetric approach, and a distance measure approach; these two approaches are described in detail below.

4.2.1 Volumetric Approach

The volumetric approach to mid-surface quality evaluation is based on comparing the volume of the solid part with a volume calculated using the mid-surface model. The solid volume of the mid-surface model can be "reverse engineered" using the surface area of the mid-surfaces and the thickness values that are associated with each mid-surface.

The volume of a mid-surface model with *i* faces can be calculated as the sum of the surface-area of each face (A_i), multiplied by its thickness value (T_i).

$$V_{mid} = \sum_{i=1}^{n} A_i T_i \qquad \textbf{(4.1)}$$

The mid-surface model volume (V_{mid}) can then be compared with the volume of the original CAD model (V_{part}), and the ratio V_{mid}/ V_{part} can be used to judge the accuracy of the mid-surface model, where an exact mid-surface gives a ratio $V_{mid}/ V_{part} = 1$. Any missing features in the mid-surface model would give a ratio of less than one, and any additional erroneous faces in the mid-surface model would give a ratio of greater than one.

Unfortunately this technique was found to be a very insensitive measure of mid-surface quality. The volume V_{mid} calculated from the mid-surface model often has small errors introduced by the mid-surface generation process. For some parts these errors have been found to be more than 10% of the total volume, which means that the errors inherent in the method can be much greater than the errors of interest caused by missing features. Figure 4.1 shows an example of a way in which volumetric errors may be introduced by the mid-surface generation process. Figure 4.1 (a) shows a solid model of an open box with five faces, Figure 4.1 (b) shows the mid-surfaces that would be generated for the part, and Figure 4.1 (c) shows the result of thickening the mid-surfaces to reconstruct the solid part. It can be seen that the reconstructed part is not identical to the original part because the mid-surfaces had to be extended during the mid-surface generation process to form a connected model, and these extension operations have changed their surface areas. The volume of the reconstructed part is also different to that of the original, and whilst in this example the errors are small, for a complicated part them may add up to a significant proportion of the part volume.

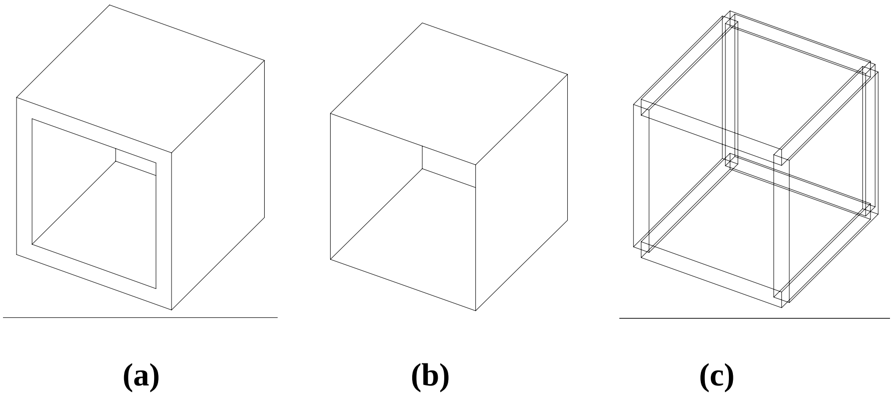


Figure 4.1 Volumetric Errors Caused by Mid-surface Generation. (a) Original Part, (b) Actual Mid-surfaces, (c) Trimmed Mid-surfaces.

A second problem with the volumetric approach is that while it may be possible to identify that a mid-surface is of poor quality, it cannot give any indication of which regions are accurately represented and which are not. The volumetric approach to evaluating the quality of the mid-surface generation was therefore rejected as not sufficiently sensitive to provide a useful measure of mid-surface quality.

4.2.2 Distance Measure Approach

The second approach that has been investigated to evaluate the quality of the mid-surface representation is a distance measure that computes the distance between points on the solid part and points on the mid-surface model. The general idea of this approach is to identify regions on the solid part that are missing from the mid-surface model by measuring the dissimilarity between points on the solid part and points on its mid-surface.

Researchers in the field of 2D image processing have commonly used distance measures to compare similar images, and these techniques are also now being used for 3D shape comparison and retrieval (Iyer et al, 2005). The Hausdorff distance is a distance measure that was introduced by Felix Hausdorff in 1918 and has been applied successfully to 2D and 3D shape similarity evaluation in the last 20 years. The Hausdorff distance measures the dissimilarity between two point sets by measuring the extent to which each point in one point set lies near some point in another point set (Huttenlocher, 1993). The Hausdorff distance has a benefit over other distance measures in that it does not require a one to one mapping between the data points in the two point sets that are being compared (Aspert, 2002).

Formally the *directed Hausdorff distance* is defined as the maximum over all the points in point set *X* of the minimum distances to point set *Y*, where *d(x,y)* is the 3D distance between *x* and *y*. (Iyer, 2005):

$$\vec{h}(X,Y) = \max_{x \in X} \min_{y \in Y} d(x,y) \qquad \textbf{(4.2)}$$

The *directed Hausdorff distance* gives different results depending on the direction of the comparison, (that is $\vec{h}(X,Y) \neq \vec{h}(Y,X)$), and the *Hausdorff distance* is therefore defined as the larger of $\vec{h}(X,Y)$ and $\vec{h}(Y,X)$.

$$H(H,Y) = \max\left\{\vec{h}(X,Y), \vec{h}(Y,X)\right\} \qquad \textbf{(4.3)}$$

Huttenlocher et al (1993) use the Hausdorff distance to determine the "degree of resemblance between two objects that are superimposed on one another". Figure 4.2 shows an example of using the Hausdorff distance to match similar outline shapes in 2D images. In the Figure 4.2 (a) is the shape to be matched, Figure 4.2 (b) is the image to match against and Figure 4.2 (c) shows the shape to match positioned and oriented on the image.

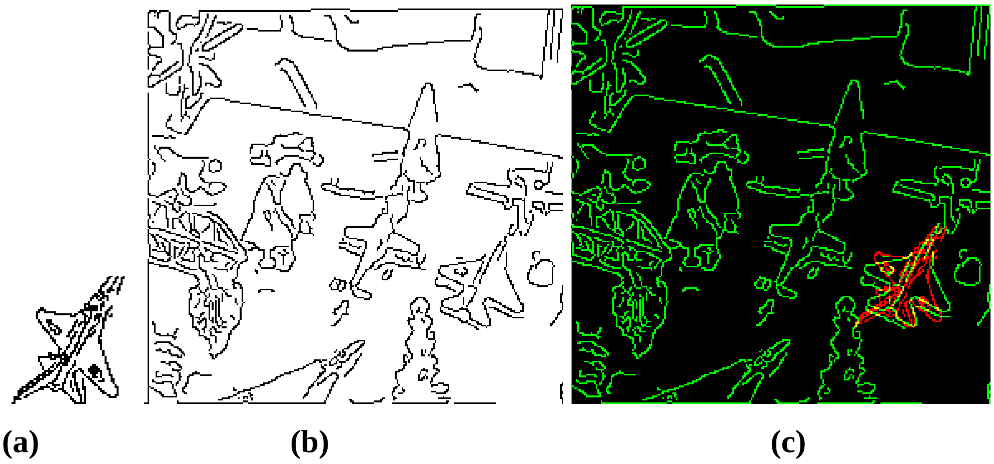


Figure 4.2 Example of Image Matching using Distance Measure (Leventon, 1995) Where (a) Is The Shape To Match (b) Is The Image and (c) Is The Shape Positioned On The Image

The Hausdorff distance has also been used for 3D shape matching (Vergeest, 2003) (Aspert, 2002), but the difficulty in applying the Hausdorff distance to 3D shapes is that it requires the geometry to be discretised into a finite set of points for comparison. A set of points must be generated on the surface of the model and the point discretisation needs to be carefully selected to achieve good results. For a large point set the technique is very computationally expensive.

In this project the Hausdorff distance has been used to measure the dissimilarity between points on the surface of a solid part and its mid-surface. In general the distance from any point on the surface of a solid part to the closest part on its mid-surface will be half the local wall thickness, with some exceptions that will be described below. Figure 4.3 shows an example for a simple X-junction part (the figure shows a cross section through the three-dimensional part). Figure 4.3 (a) shows the cross section with the solid walls represented as thick lines, and the mid-surfaces represented as thin lines, all four walls have wall thickness *t*. In Figure 4.3 (b) the minimum distance from the solid model to mid-surface is shown for two points on the solid part. Point p_1 is an arbitrary point on the surface of the solid model, and the distance to the nearest point on the mid-surface model is *t*/2. Point p2 is an example of a specific point for which the distance to the solid model is greater than t/2. In this case the closest distance from p2 to the solid model is 0.7*t*.

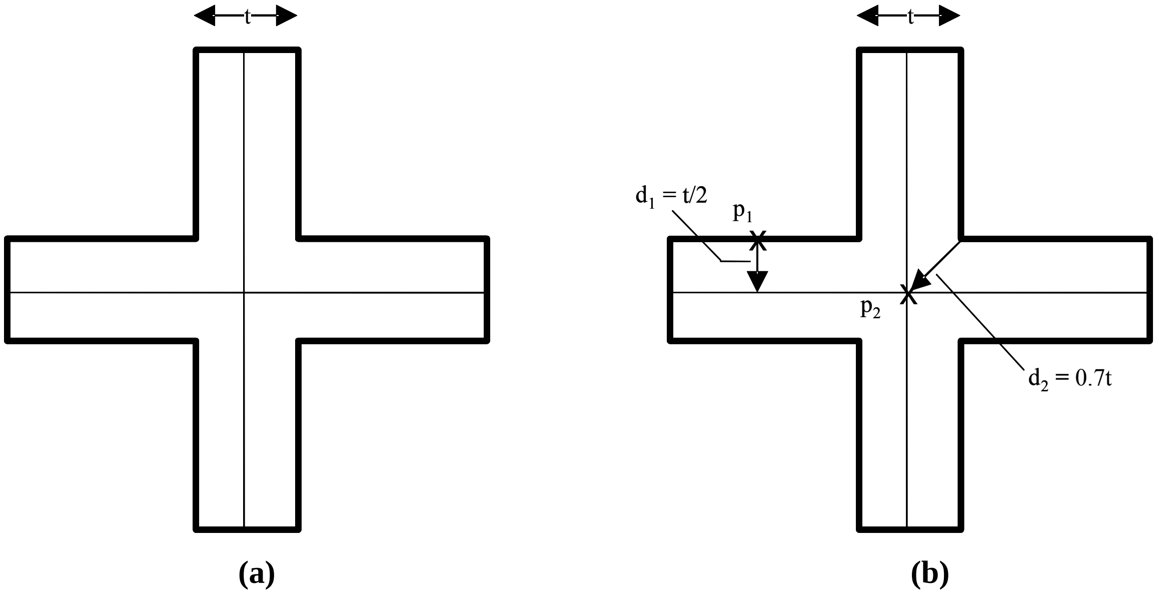


Figure 4.3 Solid Part to Mid-surface Distances

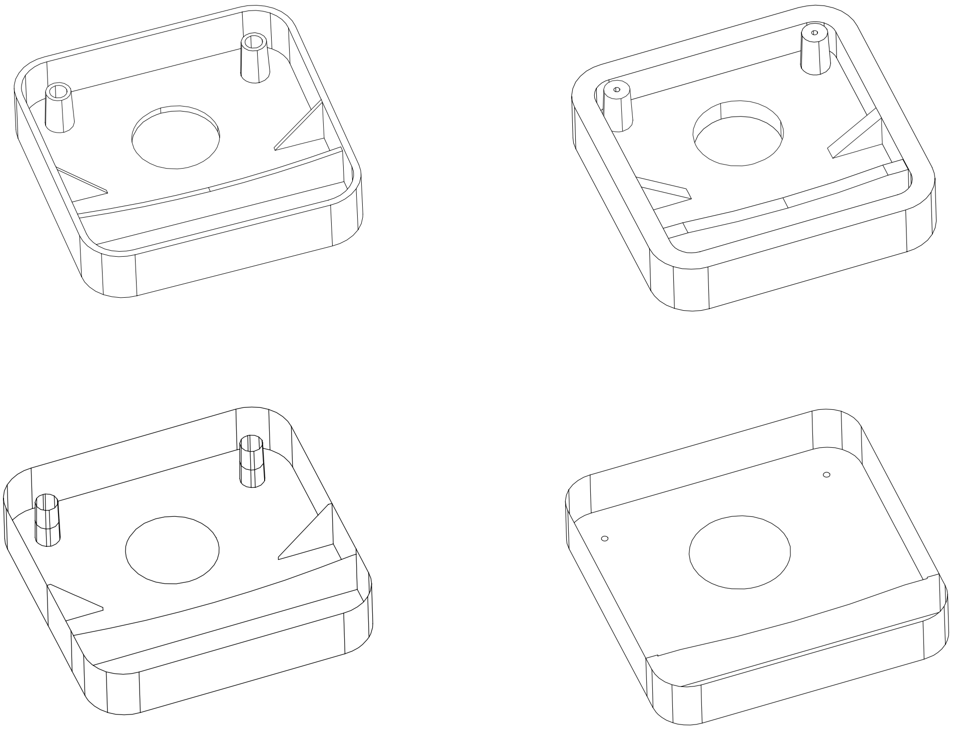
Where (a) Shows a Section Through a an X-junction and (b) Sample Distances from Mid-surface to Solid Part (d_1 and d_2)

Note that the Hausdorff distance is a directed measure. For this application it is the distance from the solid model to the mid-surface model that is of interest because the aim is to identify regions that exist on the solid model and are missing from the mid-surface model. Also, formally the Hausdorff distance is a single value that represents the maximum distance between two point sets, but the distances between all points in the two sets can be visualised by plotting all of the values $\min_{y \in Y} d(x, y)$.

Initially a simplified Hausdorff comparison was investigated that used the vertices of the two models as the basis for comparison. This approach appeared to be promising because in general the locations of the vertices on the mid-surface model are well matched to those on the solid model, however after an initial investigation it was found that there are too many exceptions which can give erroneous results so the approach was rejected.

A more complete Hausdorff comparison was then investigated by generating a grid of points on the surfaces of the two models to be compared. The density of the grid points needs to be small enough to ensure that the grid-point locations do not affect the results. For the mid-surface model evaluation the grid density must be less than the part wall thickness to ensure useful results.

In order to illustrate the proposed approach, the Hausdorff distance has been used to evaluate the mid-surface quality of two simple parts. The parts have the same shape, but one has a main wall thickness of 7.5 mm, and the other a main wall thickness of 2.5 mm. The solid geometry of the two parts, and the mid-surface abstractions generated from the parts are shown in Figure 4.4. The parts have been selected to allow the results for an accurate and an inaccurate mid-surface model to be compared.



(a) **Main Wall Thickness = 2.5 mm**

(b) **Main Wall Thickness = 7.5 mm**

Figure 4.4 Solid Geometry and Mid-Surface Models of Test Parts

For testing purposes points have been generated on the model surfaces using an automatic finite element mesh generator, although there are other methods that could also be used to discretise the models into a regular grid of points. Figure 4.5 shows an the 2.5 mm wall thickness model discretised into a grid of 3D points using the finite element mesh generator. The mesh was defined with a 2.5 mm element size which was selected as a compromise between results accuracy and computation time.

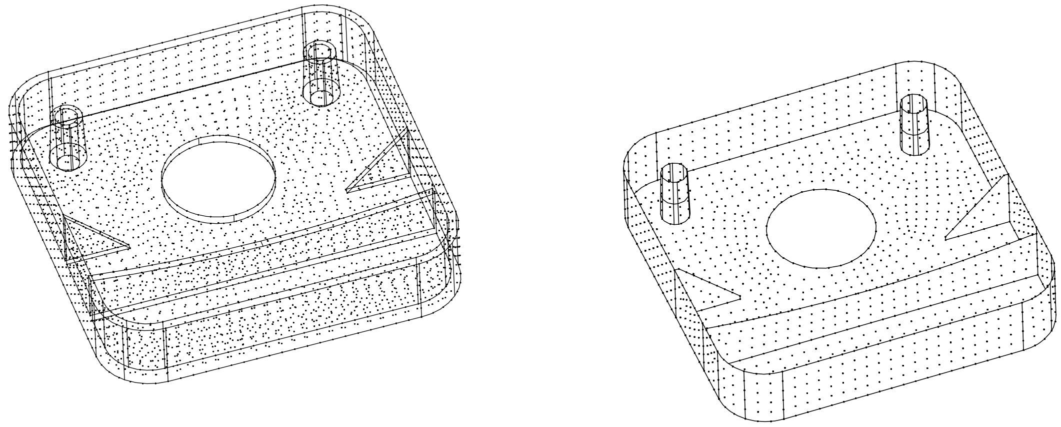


Figure 4.5 Surface Grids Generated on Test Part (2.5 mm grid spacing).

The directed Hausdorff distance has been calculated between the set of points on the surfaces of the solid model (X) and the set of mid-surface points (Y) for the part. The distance *d(x,y)* for each point has been calculated as the 3D distance between the x,y,z coordinates of point *x* and point *y*. A C++ program has been written to compute the minimum distance to any point in the mid-surface point set (Y) from each point in the solid point set (X). These values can be plotted to give a graphical representation of the minimum distance from each point on the solid model to any point on the mid-surface model. Figure 4.6 shows sample contour plots of the minimum distances for the two models sample parts shown in Figure 4.4. In the plots it can be seen that the maximum distance from a point in X to any point in Y for the thin walled part (a) is 2.8 mm, whereas the maximum distance on the thick walled part (b) is 14.5 mm indicating that some regions of the solid model are not represented on the mid-surface model.

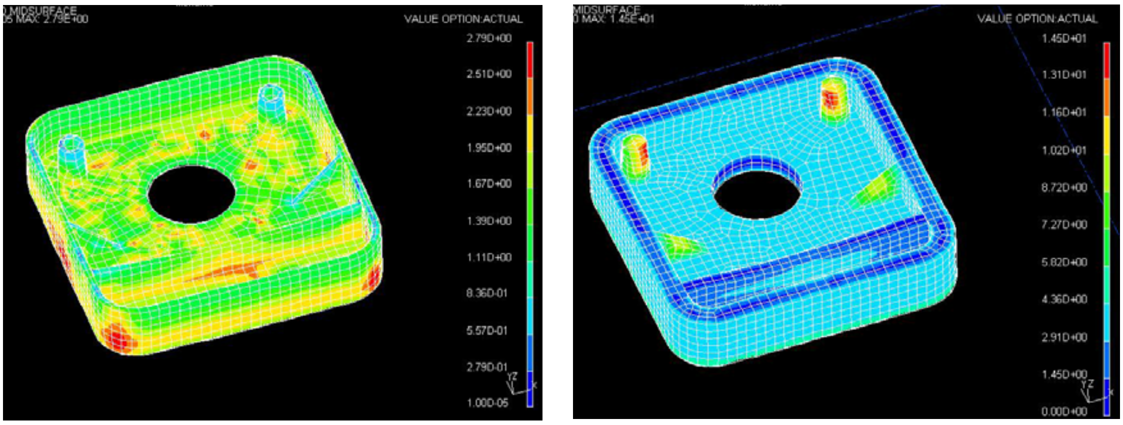


Figure 4.6 Contour Plots of Hausdorff Distance Results For Simple Parts (Main Wall Thicknesses 2.5 mm and 7.5 mm)

The results can be visualised more clearly by setting the colour contours to highlight regions where the minimum distance is greater than a specified threshold value. Figure 4.7 shows the same results plotted with only two contour levels, and the threshold value set to be 10% greater than the wall thickness. It is clear that on the thick part the points in the region of the two bosses and the two buttresses are far from the closest points on the mid-surface model, indicating that those features are missing from the mid-surface model.

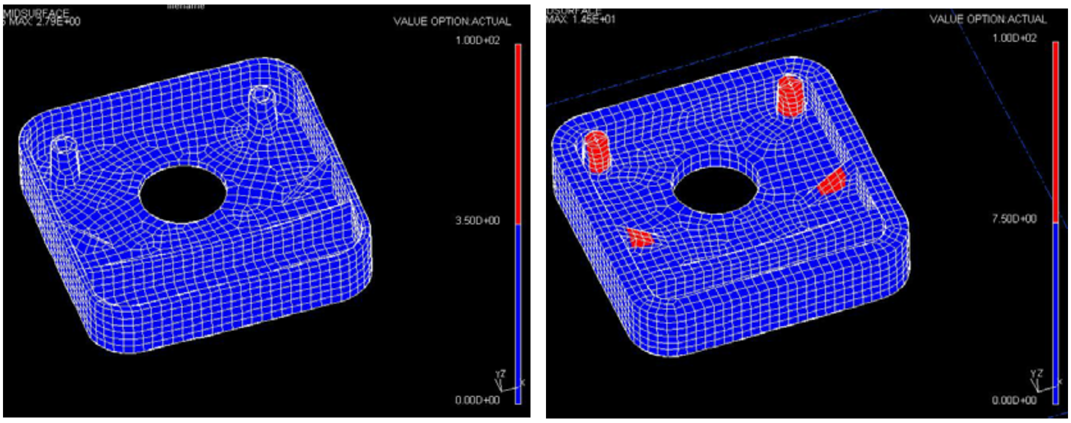


Figure 4.7 Contour Plots Of Hausdorff Distances For Simple Parts Using Threshold Value Set Wall Thickness + 10%.

A mid-surface quality factor (MQF) has been defined as the percentage of points on the solid-model that are within a specified threshold distance to any point on the mid-surface model. The selection of the threshold value has been found to be important to obtaining useful results. Initially a threshold value equal to the wall thickness was tested, but it was found that irregularities in the grid of points would sometimes falsely identify failed regions. Testing on a range of parts has suggested that a value threshold value of wall thickness + 10% gives acceptable results, although further work could be done to refine this value.

$$MQF = \frac{|X_{pass}|}{|X|} \tag{4.4}$$

Where

$|X|$ = The number of points in the set of solid model points X.

$|X_{pass}|$ = The number of points in X_{pass}, where $X_{pass} = \{X \mid X \leq T\}$

T = Threshold value for evaluation

The mid-surface quality evaluation results for the two parts shown in Figure 4.4 are presented in Table 4.1 below:

Part Name	**Simple Plastic Part (wall thickness = 2.5 mm)**	**Simple Plastic Part (wall thickness = 7.5 mm)**
Mesh size	2.5 mm	2.5 mm
Threshold value	2.75 mm	8.25 mm
MQF	0.999	0.905

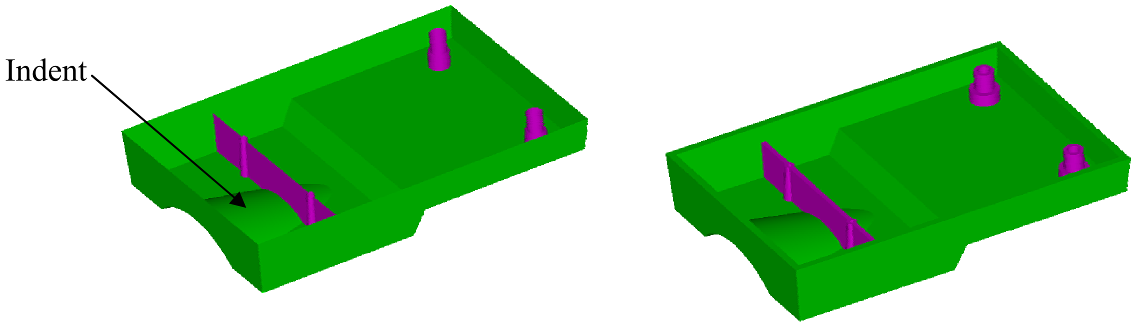
Table 4.1 Mid-surface Quality Factor Test Results

The mid-surface quality factor using the Hausdorff distance has been shown to provide a useful measure of mid-surface quality. The Hausdorff distance also provides a graphical display of the missing features in the mid-surface model. The main disadvantage of the technique is that for complicated parts with many features and a wide range of wall thicknesses it may be necessary to generate a very dense grid of points to evaluate the mid-surface quality.

4.3 Feature Recognition Results Evaluation

The mid-surface quality evaluation described in the preceding sections provides information about the quality of the mid-surface geometry prior to feature recognition. This section describes a methodology to evaluate the quality of the feature recognition results. The objective is to determine the proportion of the features in the model that have been identified by the feature recogniser. The feature recognition results are evaluated using a function that maps the number and type of features identified on the mid-surface model, to the number of faces that would be expected to represent those features in a solid model.

In general moulded parts are constructed from a main wall (representing the main shape of the part) and functional features added to or removed from the main wall. Figure 4.8 shows an example moulded part with a main wall, and attached features.



(a) Mid-surface Model **(b) Solid Model**

Figure 4.8 Example Part Showing Main Wall and Attached Features. (Main Wall Shown In Green, Attached Features Shown In Magenta)

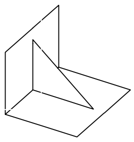
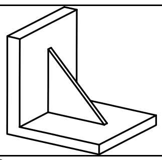
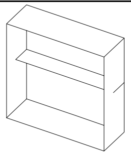
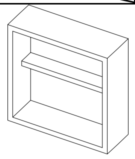
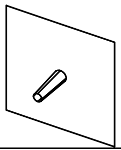
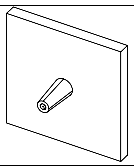
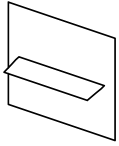
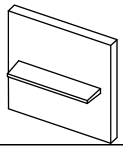
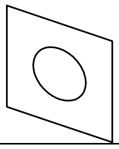
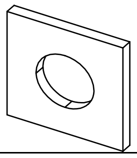
The feature recognition results evaluation methodology presented here is based on defining a mapping between the number and types of features identified in the mid-surface model, and the number of faces that would be required to represent those features in a solid model. For example the indent face shown in Figure 4.8 belongs to the main wall of the part and is represented by one surrounded face on the mid-surface model (a surrounded face is a face that is connected to other faces along all its edges). The indent face on the mid-surface model requires two faces to represent it in the solid model (an inside and an outside face) and there is therefore a 1:2 mapping between the number of surrounded main wall mid-surface faces, and the number of solid model faces required to represent those surrounded faces in a solid model. Similar mappings have been defined for all the feature and main wall types in the mid-surface model.

The mapping function allows the actual number of faces in the original part solid model to be compared with the number of faces calculated using feature recognition results and provides a measure of feature recognition quality. The evaluation is computed in two parts – a features element which is applied to faces that have been recognised as moulding features and a main wall element that is applied to faces that belong to the mid-surface main wall.

4.3.1 Mid-surface Features Evaluation

The first element of the feature recognition results evaluation considers the mid-surface faces that have been recognised as moulding features. Table 4.2 shows some examples of moulding features from the feature library and their solid model face mappings. For most isolated positive features (features that add material to the part) the mapping from mid-surface faces to solid faces is equal to the number of free edges on the feature plus two. A free edge is defined as a mid-surface edge that is not connected to any other faces, and which maps to a face in the solid part. For negative features (features that remove material from the part) the mapping is equal to the number free edges on the feature face.

For example the buttress feature shown in Table 4.2 has one free edge, so the mapping for a mid-surface face representing a buttress is calculated to be 3. The result can be verified by observation of the buttress feature solid model shown in the table, in which the buttress feature is represented by three faces (one front face, one back face, and one edge face).

Feature Type	Mid-surface Feature	Solid Feature	Solid Face Mapping
Buttress			$E_{ffree} + 2$
Part height rib			$E_{ffree} + 2$
Boss			$E_{ffree} + 2$
Fin			$E_{ffree} + 2$
Hole			E_{ffree}

E_{ffree} = number of feature free edges

Table 4.2 Solid Faces Mappings for Mid-Surface Features

The face mapping for isolated features is relatively straightforward, but it becomes more complicated when there are adjacent interacting features. Table 4.3 shows examples of exceptions to the face mapping rule caused by interactions between adjacent free edges. In the table the full height rib feature shares its top face with the top face of the part main wall, reducing the face mapping for this feature from 1:3 to 1:2. The rib group represents four intersecting ribs which share the same top face, and the one-sided rib group represents three intersecting ribs which share the same top face, and one side face.

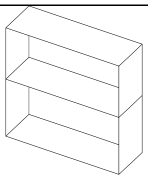
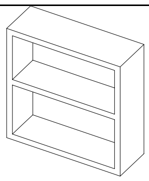
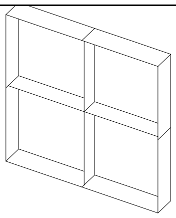
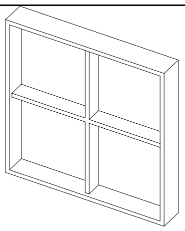
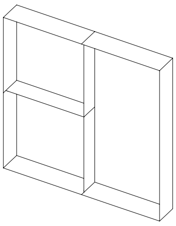
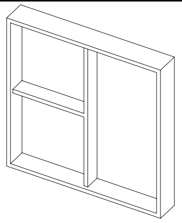
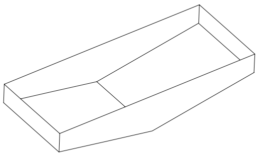
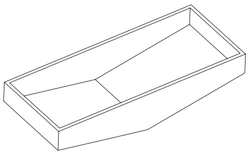
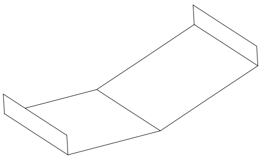
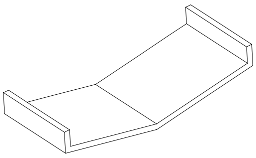
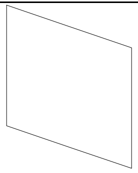
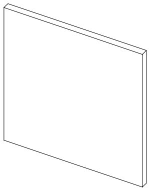
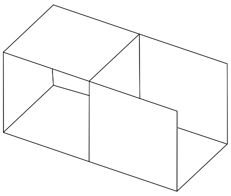
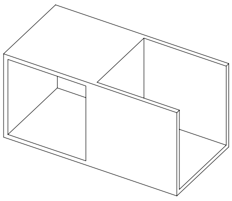
Feature Type	Mid-surface Feature	Solid Feature	Solid Face Mapping
Full Height Rib			Full height rib adds two faces to the solid model
Rib Group			Four connected ribs add nine faces to the solid model (the four top faces are merged)
One sided Rib Group			Three connected ribs add six faces to the model (three top faces are merged and two side faces merged)

Table 4.3 Examples of Interacting Mid-Surface Features

The face mappings for interacting features have been accommodated in the face mapping formula by classifying free edges into “face-generating free edges” which are free edges on the mid-surface model that generate a face on the solid model and “non-face generating free edges” which do not generate a face on the solid model. For example the free edge on the full height rib in Table 4.3 is a non-face generating free edge. An additional factor has also been defined to take into account one sided rib junctions such as the one shown in Table 4.3.

4.3.2 Main-Wall Evaluation

The second part of the feature recognition results evaluation considers the faces of the mid-surface model that have been identified as main wall faces. Due to the wide range of shapes and topologies that can be found in moulded parts it has not been possible to devise a face mapping function for all possible main wall configurations. However, three common main wall types have been defined for which it is possible to define a face mapping, and these are shown in Table 4.4. A fourth main wall classification has been defined for all other main wall geometries.

Wall configuration	Mid-surface wall	Solid wall	Solid Face Mapping
Shell			$F_{mw} + F_{cv}$
Split-Shell			$2F_{mw} + F_{cv} + F_{split}$
Plate			$2 + E_{mwfree}$
Multi-Connected			Not Defined

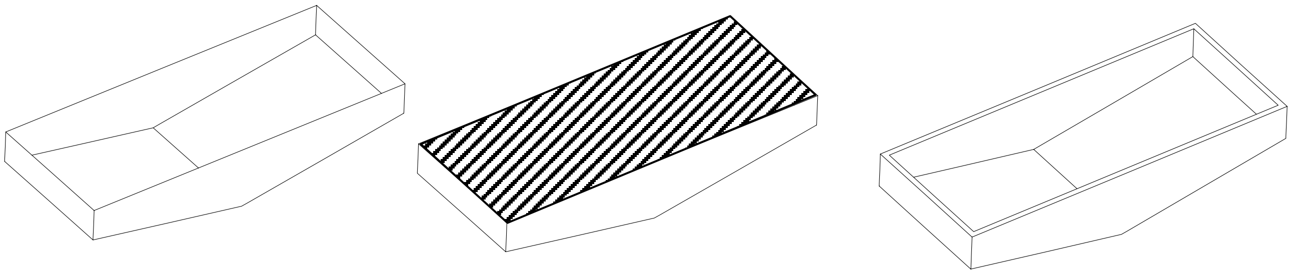
F_{mw} = number of mid-surface faces in the main wall
F_{cv} = minimum number of faces required to form a closed volume around the main wall faces
F_{split} = number of split faces in the main wall
E_{mwfree} = number of main wall free edges

Table 4.4 Examples of Moulded Part Main Wall Types

The first three configurations represent common forms for moulded parts (defined in the table as shell, split-shell and plate). A wide range of moulded parts can be classified as having a "shell" main wall, and the plate, and split-shell configurations are special cases of the shell shape. The face mappings for the three main-wall types are described in the following sections.

4.3.2.1 Main Wall - Shell

The shell main wall is defined as a part in which the main wall has a single opening (see Table 4.4). For any shell main wall it is possible to convert the shell into a closed volume by adding one or more adjacent faces to cover the opening. The mapping function for a shell main wall can be defined as the number of main wall faces in the mid-surface model plus the number of faces that would be required to form a closed volume around the mid-surface model. Figure 4.9 shows an example of the main wall mapping for a simple shell part. In Figure 4.9 (a) the mid-surface model of the part is shown with 6 faces and figure 4.9 (b) shows the closed volume formed around the mid-surface faces with one closing face (the planar closing face is shown cross hatched). The mapping function therefore calculates the number of faces that would be required to represent the mid-surface main wall in a solid model to be 13, which is equal to the number of faces in the solid part shown in Figure 4.9 (c).



$F_{mw} = 6$

(a)

$F_{cv} = 7$

(b)

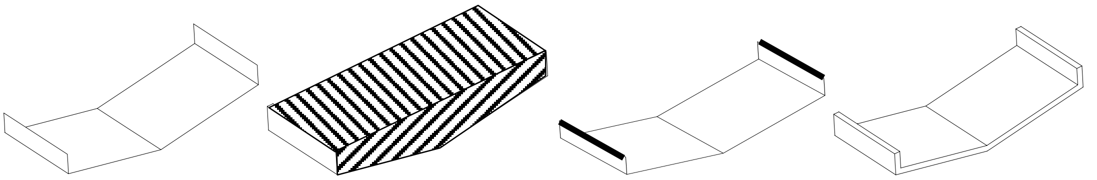
Faces in Solid Part = 13

(c)

Figure 4.9 Example of Face Mapping for a Shell Type Main Wall.

4.3.2.2 Main Wall – Split Shell

The split-shell main wall is a special case of the shell main wall in which additional faces are formed by multiple intersections between the mid-surface faces and a closing face. Figure 4.10 shows an example of a split-shell main wall. In Figure 4.10 (a) the number of mid-surface faces is 4, in 4.10 (b) the closed volume is formed around the mid-surface model with three closing faces, and in 4.10 (c) it can be seen that a split face is formed where two separate edges of the mid-surface intersect with the top closing face. The mapping function therefore calculates the number of faces required to represent the mid-surface main wall in a solid model to be 12 which is equal to the number of faces in the solid model shown in Figure 4.10(d).



$F_{mw} = 4$ **(a)**

$\mathbf{F_{cv} = 7}$ **(b)**

F_{split}=1 **(c)**

Faces in Solid Part = 12 **(d)**

Figure 4.10 Example of Face Mapping for a Split-Shell Type Main Wall.

4.3.2.3 Main Wall – Plate

The plate type main wall is a special case of a main wall formed from a single mid-surface face. The number of faces that would be required to represent the plate main wall in a solid model is calculated to be two, plus the number of edges on the face boundary.

4.3.2.4 Main Wall – Multi-Connected

If the main wall of the part is multiply connected and cannot be classified as a simple shell or plate type then it is not possible to define a face mapping between the mid-surface and solid model faces because there are too many unknowns. At present the multi-connected shell has not been considered in the feature recognition results evaluation.

4.3.3 Feature Recognition Results Evaluation Formula

A feature recognition results evaluation formula has been developed to measure the quality of the feature recognition results. The formula uses the mapping values described above to calculate the number of faces that would be required to represent the recognised mid-surface features in a solid model. At present the formula has only been developed for shell type main walls, and assumes that each recognised feature is represented by a single mid-surface face.

In addition to the factors already discussed in this chapter, an additional factor (F_{div}) has been introduced to the function to ensure correct results when using input data from the current version of the I-DEAS mid-surface function. During mid-surface generation the I-DEAS software automatically divides some mid-surface faces where they intersect with other faces. Figure 4.11 shows an example of this surface division which occurs for part-height ribs and buttresses.

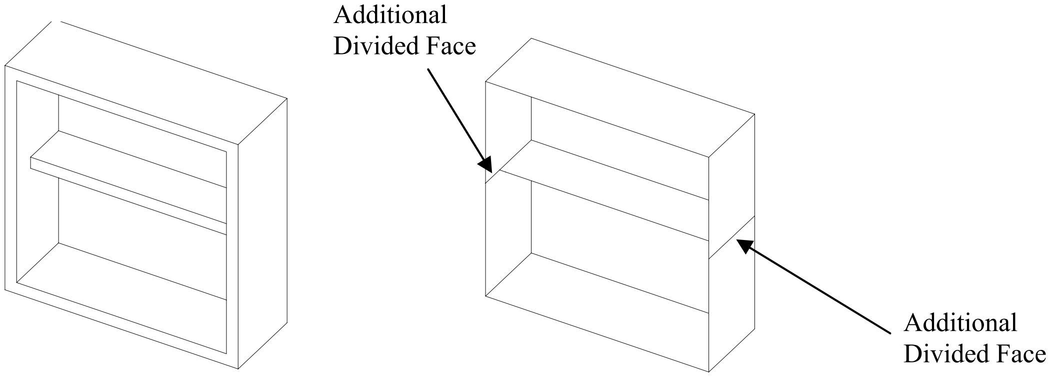


Figure 4.11 I-DEAS Mid-Surface Division at Surface Intersection

Mid-surface Face Mapping Formula for shell type parts:

$$F_{solid} = (F_{mw} + F_{cv} - F_{div}) + (2F_{pos} + E_{fgfe} - E_{oj}) \qquad \textbf{(4.5)}$$

Where

F_{solid} = Calculated number of **solid** faces
F_{mw} = Number of **main wall** faces in the mid-surface model
F_{cv} = Minimum number of faces required to form a **closed volume** around the main wall faces
F_{div} = Number of additional **divided** faces due to I-DEAS mid-surface generation
F_{pos} = Number of **positive** feature faces (bosses, fins, ribs, buttresses)
E_{fgfe} = Number of **face generating feature free-edges**
E_{oj} = Number of **one-sided rib-junction** edges

The mid-surface mapping formula can be used to evaluate the feature recognition results by comparing the calculated number of solid faces obtained using (4.5) with the actual number of faces in the original solid model of the part. The feature recognition results factor (FRF) has been defined as the number of solid faces calculated using the face mapping formula (*$F_{solid)}$* divided by the actual number of faces in the solid part (*F_{actual})*.

Feature Recognition Results Factor:

$$FRF = \frac{F_{solid}}{F_{actual}} \qquad \textbf{(4.6)}$$

Where:

F_{solid} = Calculated number of faces on solid part
F_{actual} = Actual number of faces on solid part

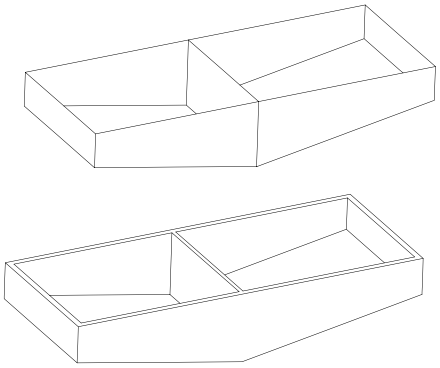
The FRF results can be interpreted as follows:

FRF = 1 If the calculated number of faces is equal to the actual number of faces then the confidence that the feature recognition has produced the correct results is high.

FRF < 1 If the calculated number of faces is less than the actual number of faces it implies that not all the features in the model were identified. This could be due to the features in the part not being within the scope of the feature recogniser, or the mid-surface being poorly defined.

FRF > 1 If the calculated number of faces is greater than the actual number of faces it implies that additional features have been incorrectly identified on the model. This is usually due to a poorly connected mid-surface

The use of the FRF will be illustrated using some simple examples. The first example is a shell type part with one rib feature shown in Figure 4.12. The formula calculates the number of solid faces required to represent the mid-surface part to be 17, which is equal to the actual number of faces in the solid part. The FRF is therefore calculated to be 1, indicating that all of the features on the part have been identified.



Feature recognition results:
1 rib and 8 main wall faces.

Number of faces in solid model (calculated using face mapping formula):

F_{solid} = (8 + 7 - 0) + (2*1 + 0 - 0)

$\mathbf{F_{solid}}$ **= 17**

Number of Faces in original solid part ($\mathbf{F_{actual}}$):
17

FRF = F_{solid}/ F_{actual} = 1

Figure 4.12 Feature Recognition Results Evaluation for Simple Shell Part

The FRF results can be seen more clearly on a more realistic test part. The feature recognition results have been evaluated for the simple plastic part previously presented in Chapter 3. In order to simulate the results for both good and bad feature recognition results the evaluation has been performed twice – first for the part with all moulding features correctly identified, and second for the part with none of the moulding features recognised (i.e. all the faces considered to be main-wall faces). The objective of the second test is to simulate a complete failure of the feature recognition process.

Figure 4.13 shows the results of the feature recognition evaluation for the correctly recognised feature model, and Figure 4.14 shows the results of the feature recognition evaluation for the model with no recognised features, and all faces belonging to the main wall.

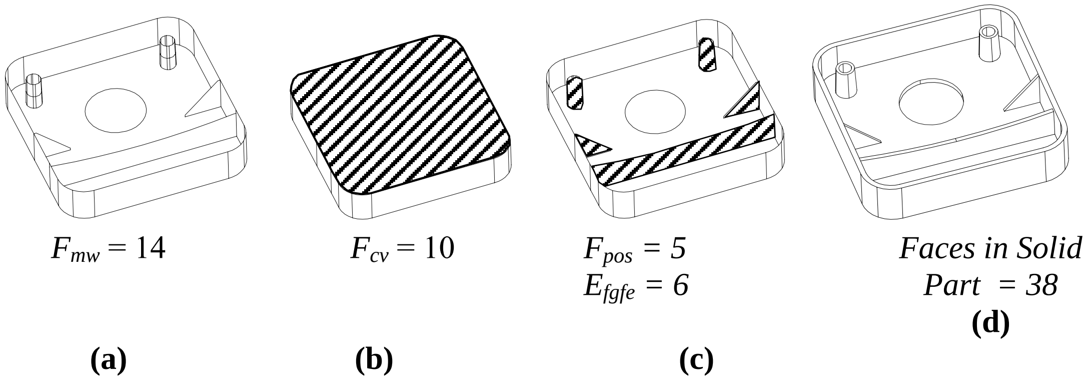
Feature recognition results:
1 rib, 2 buttresses, 2 bosses, one hole and 14 main wall faces.

Feature recognition results evaluation:

F_{solid} = (14 + 10 – 2) + (2*5 + 6 - 0)
= 38

Number of Faces in original solid part (F_{actual}):
38

FRF = F_{solid}/ F_{actual} = 1



F_{mw} = 14

F_{cv} = 10

F_{pos} = 5
E_{fgfe} = 6

Faces in Solid Part = 38

(a) (b) (c) (d)

Figure 4.13 Feature Recognition Evaluation for Correctly Recognised Model

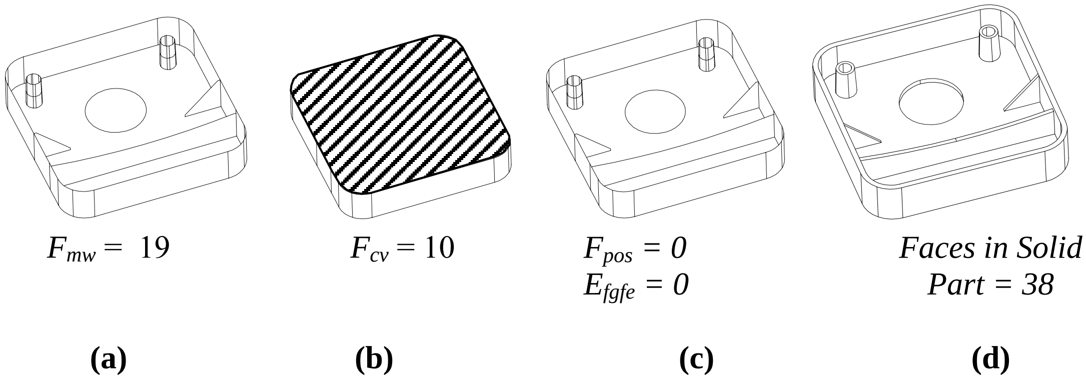
Feature recognition results:
No features recognised, 19 main wall faces.

Feature recognition results evaluation:

F_{solid} = (19 + 10 - 0) + (2*0 + 0 - 0)
= 29

Number of Faces in original solid part (F_{actual}):
38

FRF = F_{solid}/ F_{actual} = 0.76



F_{mw} = 19

F_{cv} = 10

F_{pos} = 0
E_{fgfe} = 0

Faces in Solid Part = 38

(a) (b) (c) (d)

Figure 4.14 Feature Recognition Evaluation for Poorly Recognised Model.

The FRF results provide an indication of the completeness of the feature recognition for each part. For the simple part presented in Figure 4.13 all the features have been correctly recognised and the FRF value is calculated to be 1. For the simple part presented in Figure 4.14, the FRF is calculated to be 0.76 indicating that the quality of the feature recognition is poor.

4.4 Summary and Conclusions

This chapter has presented two evaluation tools that can be used to provide feedback on the quality of the feature recognition results. The evaluation techniques provide two separate confidence measures for the feature recognition results – firstly a measure of the quality of the mid-surface (MQF), and secondly an evaluation the feature recognition results (FRF).

The MQF is able to provide a sensitive measure of the accuracy with which the mid-surface represents the original part geometry. The precision of the results is dependent on the density of the grid that is used to perform the comparison, and the limiting factor is the computational time that is required to perform the comparison.

The FRF provides a measure of completeness of the feature recognition and has been tested on a range of shell type parts. The FRF has been shown to work effectively for a range of geometries, but it is only applicable to moulded parts with a simple shell type main wall. Due to the wide range of possible moulded part geometries it has not been possible to develop a generic FRF that is applicable to all possible parts.

5 MANUFACTURING ADVISOR EXPERT SYSTEM

5.1 Introduction

This chapter describes the design of the knowledge based expert system that will generate moulding advice in the manufacturing advisor tool. The role of the expert system is to collate and apply manufacturing rules to provide tailored manufacturing advice to a designer of moulded parts. Section 5.2 provides a brief overview of expert systems design, section 5.3 defines the structure of the proposed expert system, and section 5.4 describes the knowledge elicitation process that was followed to develop the design rules for the system.

The main objective of this chapter is to define the knowledge structure for the manufacturing advisor, and to describe the knowledge acquisition and classification process that has been followed in the expert system development.

5.2 Knowledge Based Expert Systems Overview

The term knowledge based system is used to describe a computer program that stores knowledge of a particular domain and applies reasoning techniques to solve problems or give advice about the domain. An expert system is distinguished from a knowledge-based system in that it has some specialist knowledge of a realistic domain which would normally require considerable human expertise to solve, and is able to explain the reasoning for its conclusions or recommendations.

Using a knowledge-based approach gives several advantages over using a conventional sequential program. Firstly, in a knowledge-based system the knowledge is separated from the reasoning capability, which makes it easier to add and remove knowledge from the system, and to build the system incrementally. Secondly the use of a knowledge based system facilitates the encoding of heuristic rules, which may be non-deterministic in nature, and finally in the knowledge based approach the rules are encoded piecemeal and do not impose a fixed sequence of actions, which better reflects the way in which human experts solve problems.

A knowledge-based system is composed of two main elements – a knowledge base in which the facts and rules about the domain are stored, and a reasoning capability that is able to search for solutions in the knowledge domain. The facts in the knowledge base are stored using a simple grammar of symbols that encode the attributes and values for each entity in the knowledge base (often referred to as object-attribute-value triplets). The knowledge is stored as production rules which are composed of premises, and actions. If the premises for a particular rule are met, then the rule will file and perform the actions.

The rules in a knowledge-based system are not encoded in any order, and they "fire" on demand in response to inputs to the system. The knowledge-based system therefore requires a strategy to control the sequence of rule firing in the system; this capability provides the reasoning capability of the system and is referred to as an inference-engine. The role of the inference-engine is to define the search strategy adopted by the system, and control the sequence of rule firing. In particular the inference engine must have conflict resolution strategies to allow it to choose between alternative solutions when more than one rule is eligible to fire at the same time. The system should be able to inform the user of the confidence with which a result is given. (Jackson, 1999). The inference engine also controls the overall problem solving strategy adopted by the system. Two main approaches are used with production rules – either forward chaining in which the starting point is the conditions that are known to be true, and the goal is to chain forwards to find solutions that meet those initial conditions, or backward chaining in which the starting point is the goal, and the inference engine chains backwards to find initial conditions that meet the goal.

Knowledge based systems have been classified into two main types (Clancey, 1985):

- Synthetic operations that construct a system (construction):- this incorporates systems that perform design, assembly, configuration or planning tasks.

- Analytic operations that interpret a system (classification):- this incorporates systems that perform monitoring, diagnosis, control or prediction tasks.

The main difference between these two system types are that the classification system selects a solution from a set of possible existing solutions, whereas the construction system builds a solution out of more primitive existing components. In reality a practical system may have to combine elements of both approaches.

5.3 Manufacturing Advisor Expert System Design

The manufacturing advisor has been designed as a knowledge-based expert system. Manufacturing knowledge is encoded in the system as production rules, and a forward chaining strategy is used to generate appropriate manufacturing advice for features that are input to the system. The system is a "classification" expert system because it provides advice related to an existing design, as opposed to constructing a design based on design inputs.

The main requirements for the expert system are listed below. These requirements have been defined based on information on moulded part design found in design handbooks, and on discussions with practitioners in industry. The manufacturing advisor should:

(i) be able to provide designers with general manufacturing advice for a range of moulding processes and materials

(ii) be able to provide designers with manufacturing advice tailored to specific features on a designed part

(iii) be flexible enough to generate manufacturing advice for an incomplete design or a design for which the process/ or material has not been specified

(iv) be able to provide manufacturing advice at different levels of detail appropriate to different phases of the design process.

The manufacturing advisor expert system has been designed to meet these requirements through the use of a flexible knowledge representation and problem solving strategy.

Figure 5.1 shows a schematic of the knowledge representation in the expert system. In the figure it can be seen that early in the design process when there is only limited design information available the expert system will use generic moulding guidelines to generate manufacturing advice. As the design matures, and more information becomes available the manufacturing advice can be refined to a specific process, material or design priority. Finally when the CAD modelling of the part is started the manufacturing advice can be tailored to the features in the model.

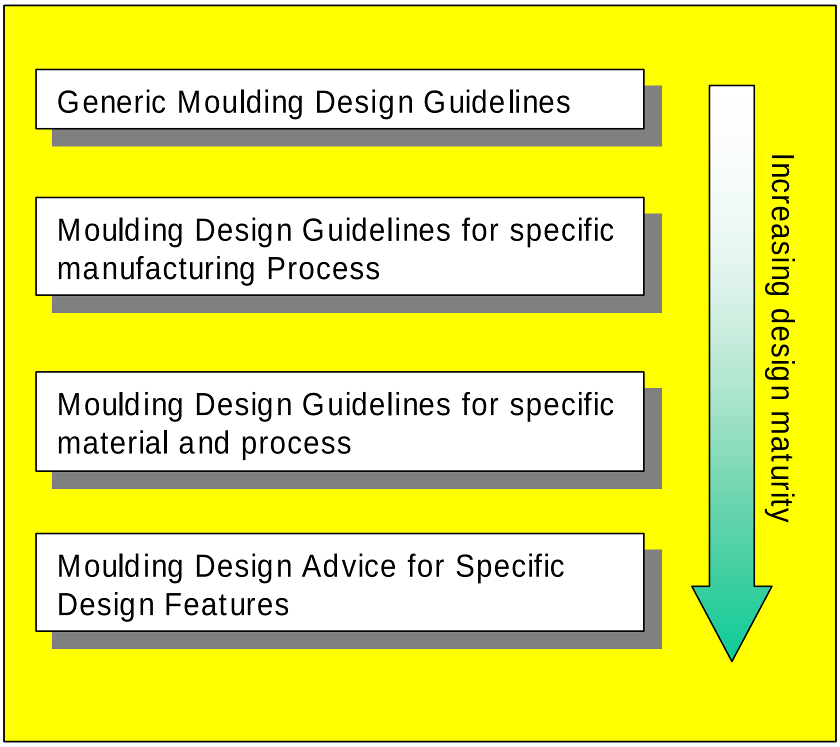

Generic Moulding Design Guidelines
Moulding Design Guidelines for specific manufacturing Process
Moulding Design Guidelines for specific material and process
Moulding Design Advice for Specific Design Features
Increasing design maturity

Figure 5.1 Schematic of Manufacturing Knowledge Representation

The expert system has been designed with two different modes of user interaction to support the different levels of design maturity. It provides an interactive mode to elicit general design information from the user through a question and answer session, and an automated file interface to read feature information that is generated by the feature recognition software.

Using a knowledge-based approach has the benefit that there is no imposed sequence on the rule firing within the system, so the system can be reactive to the user's inputs. The problem solving strategy that has been implemented uses a depth first search to identify all the available information about the part and generate relevant manufacturing advice. At each stage in an advice session the user has the opportunity to state that a particular value is "unknown" which triggers the system to try to identify a suitable value using the rules in the knowledge base.

The manufacturing advisor expert system is composed of facts about moulding processes and materials, and manufacturing rules. The following sections describe the knowledge acquisition process for the expert system. The development of the manufacturing advisor expert system demonstrator is described in chapter 6.

5.4 Knowledge Acquisition

An important aspect of expert system development is the knowledge acquisition phase. This is the task of encoding the expertise about a domain into rules in the knowledge base. Knowledge acquisition is regarded as a bottleneck of expert system development and is usually performed through extensive interviews between a knowledge engineer, and a domain expert. Jackson (1999) reports that such interviews usually elicit two to five units of knowledge (rules) per day.

In this project the manufacturing knowledge has been collated from design handbooks and written information sources, and the rule selection has been verified with a plastics design consultant at the engineering plastics manufacturer Ticona. Knowledge acquisition from design handbooks is more straighforward than elicitation from an expert because the knowledge is already encoded in a textual form, however it is still difficult to extract unambiguous rules that can be encoded in a knowledge base from the English language text.

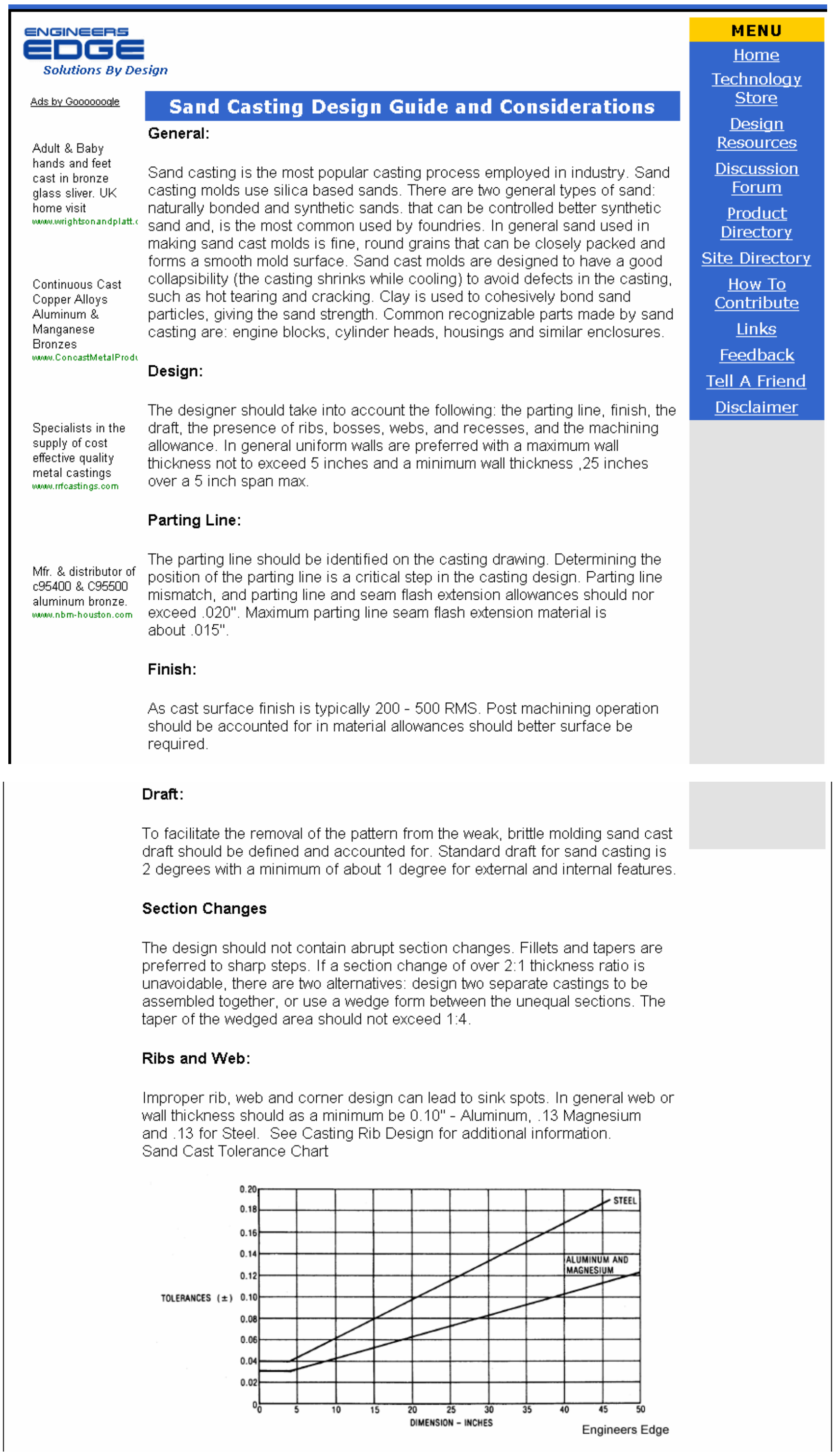
The manufacturing knowledge that has been collated has been categorised into three classes: process, material and feature knowledge, and within each class the knowledge is further categorised into rules and facts. A numbering scheme has been devised to identify each piece of manufacturing knowledge. The numbering scheme uses a two or three letter code to define the manufacturing process to which the knowledge relates, a one letter code to indicate whether the knowledge is a rule or a fact, and a number to define a unique identifier (for example SCF1 refers to fact number 1 for the sand casting process). The numbering scheme will be used to ensure that it is possible to trace the source of any advice generated by the advisor back to the original published source.

The following sections describe the rule encoding process that has been undertaken for the manufacturing advisor expert system. The knowledge that has been collated is not a complete set of design for moulding knowledge, but is a representative set that is sufficient for demonstration purposes. A complete listing of all the rules that have been encoded in the manufacturing advisor is provided in Appendix B.

5.4.1 Process Knowledge

The process knowledge encodes manufacturing rules for a particular manufacturing process, but does not take into account the specific material that will be used to manufacture the product or the detailed part design. The process rules provide the most generic manufacturing advice to the user, and will fire when there is not any more detailed design information available. The process knowledge has been generated using data from online data sources and design handbooks (Efunda, 2005, EngineersEdge, 2005, Bralla, 1986). Figure 5.2 shows a sample page from an online design guide in which the general design considerations for sand casting are presented using descriptive text (EngineersEdge, 2005) and Table 5.1 shows the rules that have been extracted from the figure. The relevant information in the design guide has been encoded as three rules, and five facts relating to the sand casting process. It is clear from this example that the process of encoding design rules from the design handbook is laborious, and not always straightforward. For example the statement “The designer should always take into account the following: the parting line, finish, the draft, the presence of ribs, bosses, webs and recesses…” does not actually contain enough specific information to form a useable design rule.

It is important that the rules are separated from the facts during the knowledge capture process. This separation of the knowledge allows the rules to be stored in a generic form, and then tailored to a specific process or material using facts for that process.



Sand Casting Design Guide and Considerations

General:

Sand casting is the most popular casting process employed in industry. Sand casting molds use silica based sands. There are two general types of sand: naturally bonded and synthetic sands. that can be controlled better synthetic sand and, is the most common used by foundries. In general sand used in making sand cast molds is fine, round grains that can be closely packed and forms a smooth mold surface. Sand cast molds are designed to have a good collapsibility (the casting shrinks while cooling) to avoid defects in the casting, such as hot tearing and cracking. Clay is used to cohesively bond sand particles, giving the sand strength. Common recognizable parts made by sand casting are: engine blocks, cylinder heads, housings and similar enclosures.

Design:

The designer should take into account the following: the parting line, finish, the draft, the presence of ribs, bosses, webs, and recesses, and the machining allowance. In general uniform walls are preferred with a maximum wall thickness not to exceed 5 inches and a minimum wall thickness ,25 inches over a 5 inch span max.

Parting Line:

The parting line should be identified on the casting drawing. Determining the position of the parting line is a critical step in the casting design. Parting line mismatch, and parting line and seam flash extension allowances should nor exceed .020". Maximum parting line seam flash extension material is about .015".

Finish:

As cast surface finish is typically 200 - 500 RMS. Post machining operation should be accounted for in material allowances should better surface be required.

Draft:

To facilitate the removal of the pattern from the weak, brittle molding sand cast draft should be defined and accounted for. Standard draft for sand casting is 2 degrees with a minimum of about 1 degree for external and internal features.

Section Changes

The design should not contain abrupt section changes. Fillets and tapers are preferred to sharp steps. If a section change of over 2:1 thickness ratio is unavoidable, there are two alternatives: design two separate castings to be assembled together, or use a wedge form between the unequal sections. The taper of the wedged area should not exceed 1:4.

Ribs and Web:

Improper rib, web and corner design can lead to sink spots. In general web or wall thickness should as a minimum be 0.10" - Aluminum, .13 Magnesium and .13 for Steel. See Casting Rib Design for additional information.
Sand Cast Tolerance Chart

Engineers Edge

Figure 5.2 Extract from Sand-casting Guidelines (EngineersEdge, 2005)

Rules extracted from guidelines in Figure 5.2		
ID	**Process**	**Rule**
GENR1	Sand Casting	Parts should be designed with a uniform wall thickness where possible
GENR5	Sand Casting	Part must be designed with adequate draft angle to facilitate removal from the mould
GENR6	Sand Casting	The design should not have abrupt section changes, where a section change is required, a gradual taper must be applied

Facts Extracted from Guidelines in Figure 5.2				
ID	**Process**	**Fact**		
SCF1	Sand Casting	Minimum wall thickness	=	6.35 mm (0.25 inches)
SCF2	Sand Casting	Maximum Wall thickness	=	127 mm (5 inches)
SCF3	Sand Casting	Main wall draft angle	=	2°
SCF4	Sand Casting	Attached Features draft angle	=	1°
SCF5	Sand Casting	Section Change Taper Ratio	=	4:1

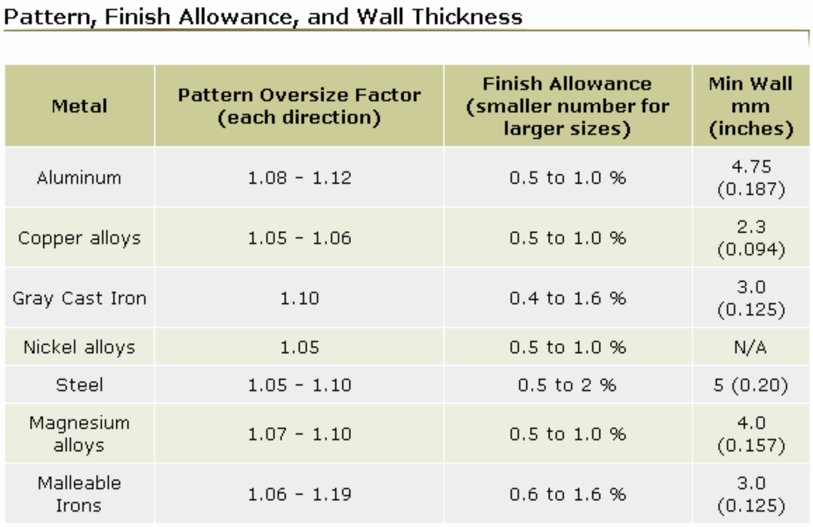
Table 5.1 Extracted Design Rules and Facts for Sand Casting

5.4.1.1 Materials Knowledge

Materials knowledge refers to the manufacturing information that is specific to a particular material. The materials knowledge allows the manufacturing advice that is generated by the system to be tailored to the specific materials. In most cases the materials knowledge only defines material specific design parameters (facts) for existing process rules.

The materials knowledge has been collated from design handbooks published by thermoplastic suppliers and other general design handbooks (Ticona, 2000, GE Plastics, 1997, Efunda, 2005, Boothroyd, Dewhurst and Knight, 2002).

Figure 5.3 provides a sample from an online design guide in which the minimum wall thickness for a variety of sand casting materials is defined in a tabular format, and Table 5.2 shows the materials knowledge that has been extracted from the table.



Pattern, Finish Allowance, and Wall Thickness

Metal	Pattern Oversize Factor (each direction)	Finish Allowance (smaller number for larger sizes)	Min Wall mm (inches)
Aluminum	1.08 - 1.12	0.5 to 1.0 %	4.75 (0.187)
Copper alloys	1.05 - 1.06	0.5 to 1.0 %	2.3 (0.094)
Gray Cast Iron	1.10	0.4 to 1.6 %	3.0 (0.125)
Nickel alloys	1.05	0.5 to 1.0 %	N/A
Steel	1.05 - 1.10	0.5 to 2 %	5 (0.20)
Magnesium alloys	1.07 - 1.10	0.5 to 1.0 %	4.0 (0.157)
Malleable Irons	1.06 - 1.19	0.6 to 1.6 %	3.0 (0.125)

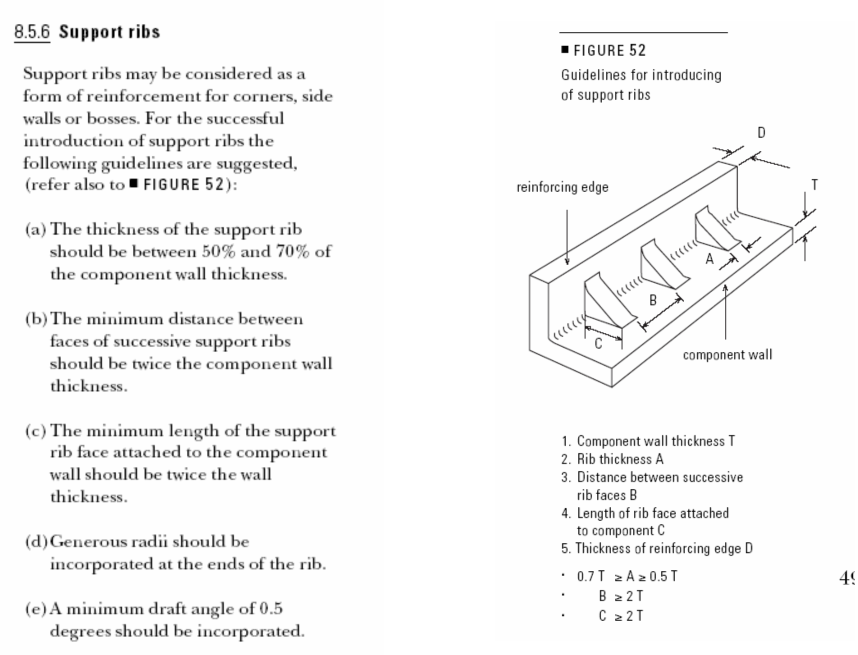
Figure 5.3 Minimum Wall Thicknesses for Sand Casting (Efunda, 2005)

Facts Extracted from Guidelines in Figure 5.3			
ID	**Process**	**Material**	**Fact**
SCF9	Sand Casting	Aluminium	Minimum wall thickness = 4.75 mm
SCF10	Sand Casting	Copper Alloys	Minimum wall thickness = 2.3 mm
SCF11	Sand Casting	Gray Cast Iron	Minimum wall thickness = 3.0 mm
SCF12	Sand Casting	Steel	Minimum wall thickness = 5.0 mm
SCF13	Sand Casting	Magnesium Alloys	Minimum wall thickness = 4.0 mm
SCF14	Sand Casting	Malleable Irons	Minimum wall thickness = 3.0 mm

Table 5.2 Facts Extracted from Figure 5.3

5.4.1.2 Feature Knowledge

Feature knowledge encodes the manufacturing knowledge for individual design features of a part. The feature rules only fire if a feature model is available to the system, and the rules are applied to every relevant feature in the model. Manufacturing advice for design features is provided in design handbooks alongside other more general design guidelines. Figure 5.4 shows a sample feature level design guideline from a plastics design guide (GE Plastics,1997), and Table 5.3 shows the rules that have been extracted from the figure. The guidelines relate to a stiffener feature which is described as a "support rib" in the figure, but is a buttress in the nomenclature of this project.



8.5.6 **Support ribs**

Support ribs may be considered as a form of reinforcement for corners, side walls or bosses. For the successful introduction of support ribs the following guidelines are suggested, (refer also to ■ FIGURE 52):

(a) The thickness of the support rib should be between 50% and 70% of the component wall thickness.

(b) The minimum distance between faces of successive support ribs should be twice the component wall thickness.

(c) The minimum length of the support rib face attached to the component wall should be twice the wall thickness.

(d) Generous radii should be incorporated at the ends of the rib.

(e) A minimum draft angle of 0.5 degrees should be incorporated.

■ FIGURE 52

Guidelines for introducing of support ribs

1. Component wall thickness T
2. Rib thickness A
3. Distance between successive rib faces B
4. Length of rib face attached to component C
5. Thickness of reinforcing edge D

- 0.7 T ≥ A ≥ 0.5 T
- B ≥ 2 T
- C ≥ 2 T

Figure 5.4 Extract from Injection Moulding Design Guidelines (GE Plastics, 1997)

Rules Extracted from Guidelines in Figure 5.4		
ID	**Process**	**Rule**
IMR1	Injection Moulding	Projections from the main wall should be designed with a proportionally smaller wall thickness than the main wall
IMR4	Injection Moulding	Buttresses must be designed with appropriate spacing
IMR5	Injection Moulding	The length of the buttress attachment face must be designed with regard to the main wall thickness
IMR6	Injection Moulding	Ribs must have generous radii
IMR7	Injection Moulding	Ribs must be designed with appropriate draft angle
Facts Extracted from Guidelines in Figure 5.4		
ID	**Process**	**Fact**
IMF7	Injection Moulding	Buttress Thickness = 50 - 70% of main wall thickness
IMF3	Injection Moulding	Minimum buttress spacing = 2 x main wall thickness
IMF4	Injection Moulding	Minimum attached edge length = 2 x main wall thickness
IMF6	Injection Moulding	Minimum Draft Angle = 0.5°

Table 5.3 Facts and Rules Extracted from Figure 5.4.

5.5 Summary and Conclusions

This chapter has described the structure of the manufacturing advisor expert system and the knowledge acquisition process that has been used to capture and classify manufacturing knowledge. The manufacturing advisor has been designed as a knowledge-based expert system in which the manufacturing knowledge is encoded using heuristic rules. The system uses a forward chaining strategy to perform the manufacturability evaluation and generate manufacturing advice. It has a hierarchical knowledge structure which allows manufacturing advice to be generated to different levels of detail, dependent on the information available to it.

The knowledge acquisition has been performed from design handbooks and online information sources. A classification scheme has been defined to identify each knowledge element, and to ensure that the published source of each item can be easily identified from the advisor outputs. A representative sample of manufacturing knowledge has been extracted and encoded for use in the expert system demonstrator. The development of the expert system demonstrator is described in Chapter 6, Section 6.3.

6 DEVELOPMENT OF MANUFACTURING ADVISOR DEMONSTRATOR

6.1 Introduction

This chapter describes the development of a demonstrator for the manufacturing advisor. The demonstrator will be used to evaluate the methodologies that have been presented in this thesis.

The manufacturing advisor is composed of two linked modules: a feature recogniser and a knowledge based expert system. The evaluation tools presented in Chapter 4 have not been implemented in the demonstrator, but will be applied manually to the test cases in Chapter 7. A flow chart of the top level manufacturing advisor architecture is shown in Figure 6.1.

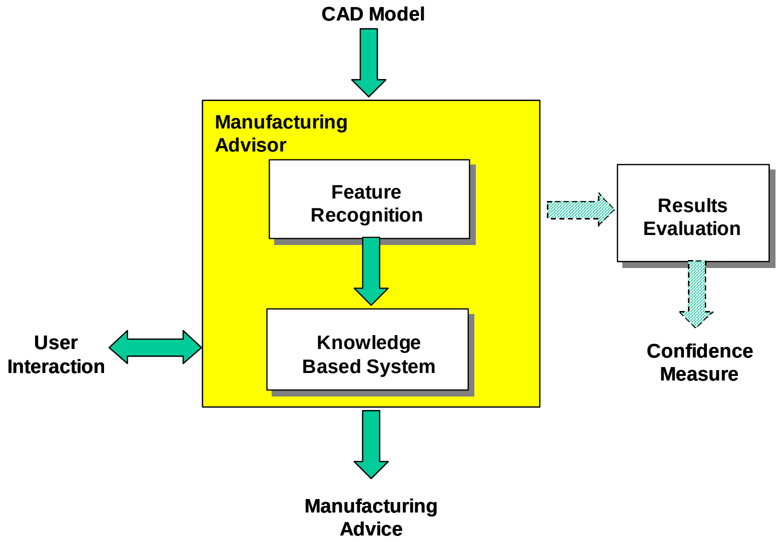


Figure 6.1 Top Level Flow Chart of Software Demonstrator

The overall philosophy for the development of the demonstrator has been to use open source software and international standards where possible. This philosophy is partly driven by a desire to develop a flexible tool that can be easily modified to work with a variety of CAD systems and computing environments, but also by a practical need to minimise the cost of implementing the demonstrator. The two main modules are described in detail in the following sections.

6.2 Feature Recognition Module

6.2.1 Overview

The purpose of the feature recognition module is to identify moulding features from a CAD solid model and allow the user to visualise the recognised features. The feature recognition module also needs to output the features in an appropriate format for use in the knowledge base. The feature recognition methodology has already been described in detail in Chapter 3.

The overall structure of the feature recognition module is shown in Figure 6.2. The input to the feature recognition module is a B-Rep solid model of the part generated using a CAD solid modeller. The first stage in the feature recognition process is a pre-processing step in which a mid-surface representation of the part is generated and exported as a STEP AP203 file using a commercial mid-surface generation tool. A text file containing the thickness values for each mid-surface face is also generated during the mid-surface generation process. The second step of the feature recognition process is to read the mid-surface data into an object-oriented data structure that can be parsed by the feature recognition algorithms. An attributed adjacency matrix is then constructed to facilitate the feature search and the feature recognition is performed. The feature recognition uses an algorithmic approach, and is able to recognise a range of common moulding features categorised into a number of feature classes. Finally the results are presented to the user in a graphical form and are also output in an interface file for input to the knowledge-based expert system.

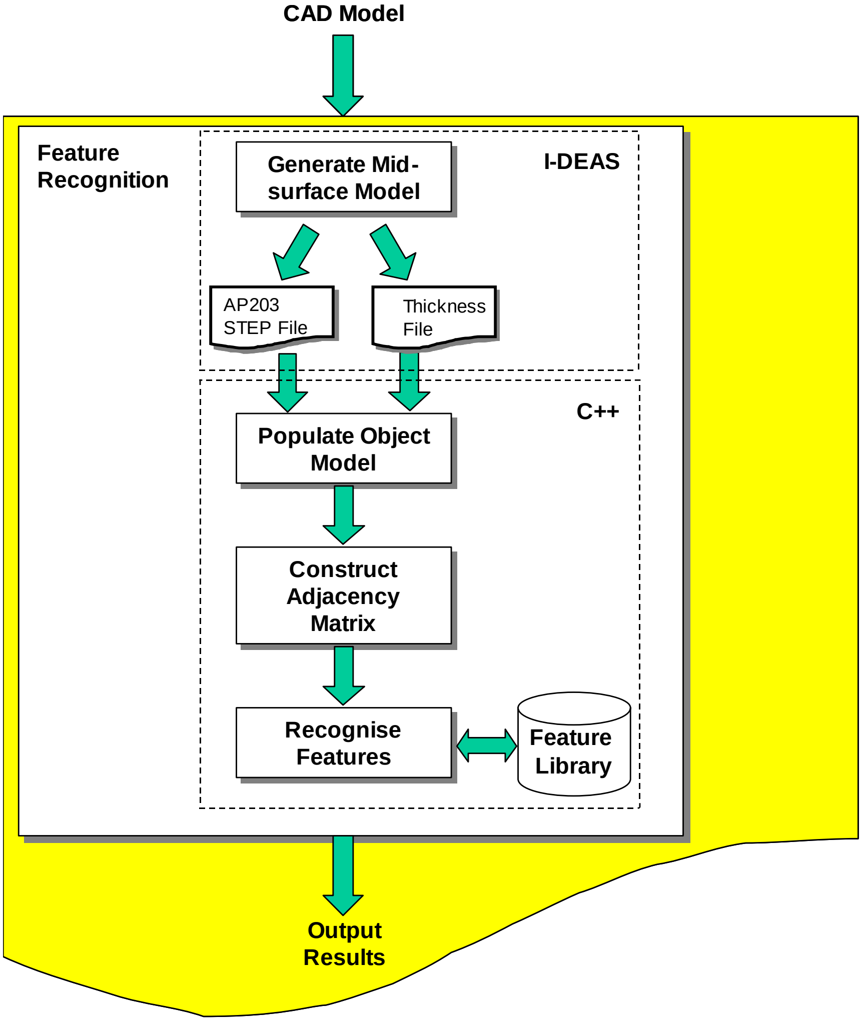


Figure 6.2 Flow Chart of Feature Recognition Process

6.2.2 Software Development Environment

The feature recognition module has been implemented using GNU C++ on the Cygwin platform. Cygwin is a free Linux-like environment for Windows (Cygwin, 2004) that allows unix software and libraries to be executed from a Windows PC. Cygwin was used for the software development to allow the feature recognition software to make use of unix based C++ libraries, whilst also being integrated with the CAD system and knowledge shell running on a Windows PC.

The CAD system UGS I-DEAS NX 11 has been used to generate the mid-surfaces in the pre-processing step of the demonstrator. There are a number of CAD systems on the market that can generate a mid-surface representation from a CAD solid model, and it would be straightforward to replace I-DEAS with an alternative tool. The data exchange between I-DEAS and the feature recognition module uses a STEP AP203 file to make the software as generic as possible, although a small amount of I-DEAS specific programming has been required to output the wall thickness values from I-DEAS.

The object-oriented mid-surface model and the feature recognition software have been developed using object-oriented C++, and the STEP Class Libraries from NIST (National Institute of Standards and Technology) have been used to develop the interface to STEP AP203 (NIST, 2002).

6.2.3 Program Structure

The feature recognition module is composed of five main sub-programs:

i. **CAD interface**

   Reads the mid-surface geometry and topology from the STEP AP203 file and interfaces to the CAD system to read wall thickness values

ii. **Object-oriented mid-surface model**

   Defines an object oriented mid-surface data structure to store the mid-surface geometry and topology prior to feature recognition

iii. **Adjacency matrix**

   Creates the mid-surface adjacency matrix as a three dimensional array using data from the mid-surface model

iv. **Feature recognition**

   Performs the feature recognition using the adjacency matrix and mid-surface model

v. **Results output**

   Generates a graphical display of the results, and outputs the results to the knowledge base.

The feature recognition programs are described in more detail in the following sections, and the source code is included in Appendix C.

6.2.3.1 CAD Interface

In order to achieve a generic basis for the feature recognition process the data exchange with the CAD system has been implemented using a STEP AP203 Part 21 file. The STEP application protocol AP203 "Manifold Surfaces with Topology" is primarily used for the exchange of manifold B-Rep solid models and surfaces without topology, but it does also provided limited support for non-manifold geometries through the open_shell entity type.

The mid-surface geometry is stored in the AP203 file as a collection of faces connected along shared edges. The mid-surface geometry is non-manifold because the model topology can contain edges that are adjacent to two, three or more faces – whereas in a manifold solid model each edge is adjacent to exactly two faces. STEP AP203 uses an eight-level hierarchy to represent the CAD geometry comprising open_shell, shell_based_surface_model, face_surface, face_bound, edge_loop, oriented_edge, edge _curve and vertex_point entities. The STEP AP203 hierarchy is shown in Figure 6.3.

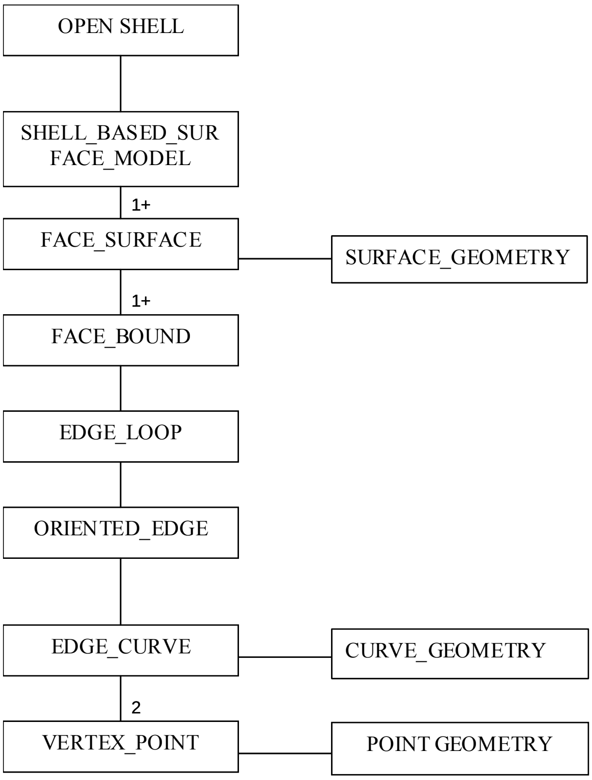


Figure 6.3 STEP AP203 Class Hierarchy

An interface program has been developed to extract the topology and geometry information that will be required for feature recognition from the STEP AP203 file. The interface program has been written in C++ and uses functions provided by the STEP Class libraries from NIST (NIST, 2002) to read entities from the STEP file. The program cycles through all the entities in the STEP file and writes the relevant entities and attributes into the object oriented mid-surface model. The mid-surface model structure is described in section 6.2.3.2.

A second interface program has been written to extract the wall thickness information from the I-DEAS CAD model. The wall thicknesses are computed automatically when the mid-surface model is generated, but the values are stored in I-DEAS and cannot be exported with the mid-surface geometry in the STEP file. An I-DEAS macro has been written to export the thickness values for each face to a text report, and a C++ program is used to extract the values from the text report and write them to the mid-surface model. At present the interface that has been developed is fairly simplistic and each wall is represented with a single uniform thickness, however the interface could be modified in the future to store the variation in wall thickness across each mid-surface.

6.2.3.2 Object-Oriented Mid-Surface Model

An object-oriented mid-surface model has been defined to store the geometry and topology information that is read from the STEP file. The mid-surface model is used only to facilitate feature recognition and is not intended to replicate the entire geometry description in the STEP file. The mid-surface model therefore stores only selected geometric attributes that are relevant for feature recognition, and does not store the complete geometry description of the surfaces and edges in the model. The mid-surface model uses six classes to represent the model hierarchy: *Midsurface Model*, *Face*, *Bound*, *Edge_Loop*, *Edge* and *Vertex*. The object model diagram for the data structure is shown in Figure 6.4 (for clarity only selected attributes are shown in the figure) and the main attributes and operations for each object class are shown in Table 6.1.

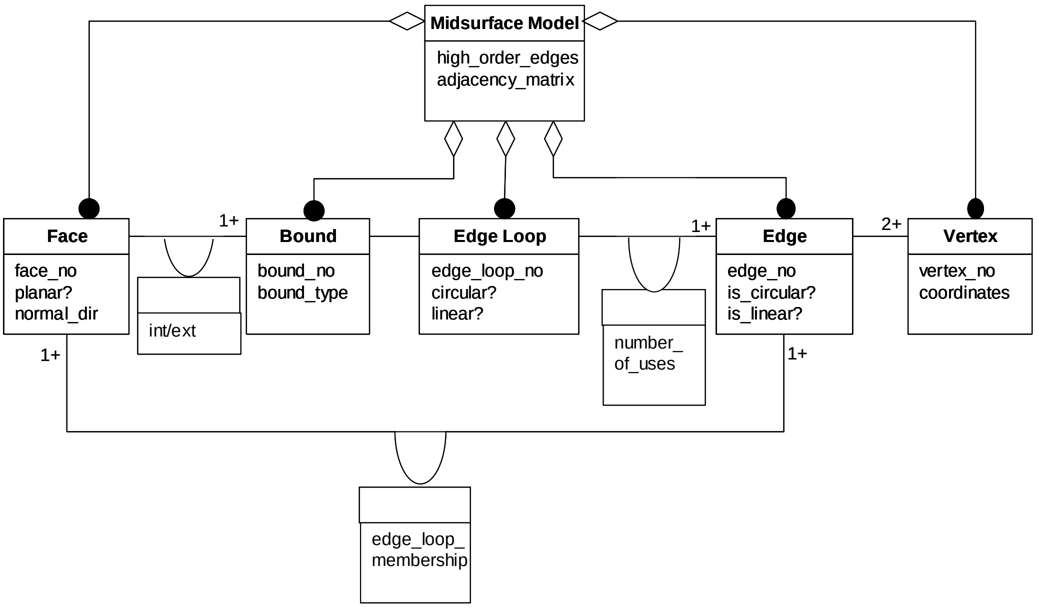


Figure 6.4 Object Model Diagram For Mid-Surface Model (Showing Selected Attributes Only)

Vertex	Edge	Edge_Loop
vertex_id coordinates	edge_id list_of_loops [ ] no_of_loops list_of_faces[ ] no_of_faces	edge_loop_id no_of_edges list_of_edges [ ] attached_faces [ ]
add_vertex get_vertex_id get_coord_component	add_edge get_edge_id add_loop_to_edge get_no_of_loops add_face_to_edge get_no_of_faces_using_edge	add_edge_loop get_edge_loop_id add_edge_to_loop get_no_of_edges get_edge add_attached_face get_no_of_attached_faces

Bound	Face	MidSurface
bound_id edge_loop_id bound_type	face_id list_of_bounds[ ] list_of_adjacent_faces[ ] planar? normal_dir	model_id adjacency_matrix [ ] faces[ ]
add_bound get_bound_id add_loop_to_bound get_bound_id get_edge_loop_id	add_face get_face_id add_bound add_face_geometry set_planar get_planar set_normal_dir get_normal_dir_component	add_shell add_edge_relations update_adj_matrix print_adj_matrix get_face_angles find_face_features find_stiffeners generate_step_overlay

Table 6.1 Object Classes, Attributes And Operations for Mid-Surface Model

The mid-surface model has been implemented using object-oriented C++. The classes shown in Table 4.1 have been programmed as C++ classes, and the operations are implemented as C++ methods. The data structure is populated by the CAD interface programs that have already been described in section 6.2.3.1.

6.2.3.3 Adjacency Matrix

The attributed mid-surface adjacency matrix is constructed using geometry and topology information from the mid-surface model. As previously described in Chapter 3 the adjacency matrix stores three attributes associated with each face-edge relation which are:

i. Edge Loop Number - specifies which edge loop contains the edge on the face.
ii. Internal/ External – specifies whether the edge belongs to an internal or external face bound
iii. Number of Uses – specifies the number of times the edge is used in the face

The adjacency matrix is stored as a three dimensional array of faces, edges and attributes. A C++ program has been written to populate the adjacency matrix by cycling through the entities in the mid-surface model. The main algorithm that is used to populate the adjacency matrix is shown below:

```
Algorithm Populate_Adjacency_Matrix
      begin
         Initialise ADJ_MATRIX [ ] [ ] [ ]
         for each face f in model do
            for each bound b in face f do
               get edge loop el for bound b
               for each edge e in el do
                  count no. of times e is used in el and store as ne
                  ADJ_MATRIX [f] [e] [0] = el
                  ADJ_MATRIX [f] [e] [1] = b
                  ADJ_MATRIX [f] [e] [2] = ne
               end
            end
         end
      end
```

6.2.3.4 Feature Recognition

The feature recognition software has been written in object oriented C++ and makes use of the adjacency matrix and mid-surface model to perform the feature recognition. The main feature recognition algorithms have already been presented in Chapter 3.

The first step in the feature recognition process is to identify the face features (features that are connected to the interior of a face). The face feature recognition algorithm cycles through all the faces in the model and identifies the faces with internal bounds. The algorithm then checks the edge connectivity for every edge in each internal bound, and uses the results to determine whether the face feature is a hole, or has attached faces. If the face feature has attached faces then further geometric checks are then performed to identify the type of attached feature. For example a face feature that is connected to the main part through a single linear edge is recognised as a fin, and a face feature with a cylindrical face that is connected to the main wall through a circular edge is recognised as a boss.

The second step in the feature recognition process is to identify the stiffener features. The stiffener feature recognition algorithm cycles through all the faces in the model, and for each face it cycles through all the edges in the face outer bound. The algorithm checks the edge order for each edge in the outer bound, and looks for the patterns of high order, and unconnected edges that characterise stiffener features. For example a face with two adjacent high order edges, and one unconnected edge would be recognised as a buttress feature.

The demonstrator performs some geometric checks to complete the feature recognition process, but these checks could be extended in future work. For example the angles between the intersecting faces of high-order junctions can be important for moulding design, but are not fully evaluated in the feature recognition demonstrator. At present the demonstrator checks the angles between the faces of 2nd order junctions and can differentiate between L and V junctions, but this functionality has not been implemented for other high order junction types such as T and X junctions.

6.2.3.5 Results Output

The feature recognition results are displayed using the results output programs. The output programs generate the results in three separate formats: a log file containing a list of the recognised features, an interface file that can be read by the expert system shell and a STEP file containing graphical representation of the results.

The log file is written by the feature recognition program and contains a list of the features that have been recognised to allow the user to make a quick check of the results. The file also lists the STEP identifiers for the mid-surface faces associated with the features to allow the user to cross reference the results back to the CAD model if required.

The interface file is written using the syntax of the CLIPS expert system shell and contains a list of features and attributes. A C++ program has been developed to output each feature to the CLIPS format file with four attributes representing the feature ID number, the feature type, the STEP ID, and the feature wall thickness. A sample interface file is shown in Figure 6.5.

```
(feature (feat-number 1) (feat-type rib) (face-id F11) (has-
thickness YES)(thickness 2.0))
(feature (feat-number 2) (feat-type rib) (face-id F12) (has-
thickness YES) (thickness 2.0))
(feature (feat-number 3) (feat-type buttress) (face-id F13)
(has-thickness YES) (thickness 3.0))
(feature (feat-number 4) (feat-type boss) (face-id F14) (has-
thickness YES) (thickness 4.0))
(feature (feat-number 5) (feat-type rib) (face-id F15) (has-
thickness YES) (thickness 1.0))
```

Figure 6.5 Sample CLIPS Interface File

The STEP output file contains a colour coded graphical representation of the feature recognition results to allow the results to be displayed using a CAD system. The STEP output file is generated using a C++ program which overlays colours onto the face entities in the mid-surface model based on the feature recognition results. The STEP output program uses a STEP entity called OVER_RIDING_STYLED_ITEM which allows additional display information to be added to entities in a STEP file.

The STEP output program writes a segment of STEP Part 21 file containing the colour overlays, and the STEP file segment is then appended to the original STEP mid-surface file using an AWK script.

6.3 Knowledge Based System Module

6.3.1 Overview

This section describes the implementation of the knowledge based system module for the demonstrator, the structure of the knowledge based expert system has already been described in Chapter 5 “Manufacturing Advisor Expert System”. The purpose of the module is to demonstrate a capability to generate manufacturing advice for moulded parts based on their geometric shape, the selected moulding process, and the specified material. The knowledge base has been structured to allow two modes of operation:

1. An interactive tool that provides manufacturing advice based on general parameters such as manufacturing process, material, wall thickness etc.
2. An automated tool that is able to provide manufacturing advice tailored to a specific geometry model by reading design information from a feature model.

The two modes of operation are tightly integrated together to allow a designer to interact with the system providing design information at increasing levels of detail as the design progresses. The source code for the manufacturing advisor expert system is provided in Appendix C.

6.3.2 Software Development Environment

The knowledge base has been implemented using the knowledge based system shell CLIPS. CLIPS (C Language Integrated Production System) is an expert system development environment that was originally developed by NASA’s Johnson Space Center in the 1980s and is now independently maintained and distributed as public domain software. CLIPS provides a tool for representing a wide range of knowledge using rule based, object oriented and procedural programming capabilities. CLIPS provides an expert system shell, with an integrated editor and debugging tool and is also available in a Windows version with a menu driven graphical user interface.

(Giarrantano, 2002). CLIPS was selected as the software development environment because it is widely used as a KBS research tool, and it is public domain software. (CLIPS, 2005)

6.3.3 Manufacturing Advisor Architecture

The manufacturing advisor expert system has been developed as a standalone application using CLIPS, and the module architecture is shown in Figure 6.6. The inputs are either entered into the system by the user during an interactive session (labelled User Interaction in the figure) or read from a feature file generated by the feature recognition module (labelled Feature Model in the figure). The output from the knowledge base is a text report containing the manufacturing advice for the session.

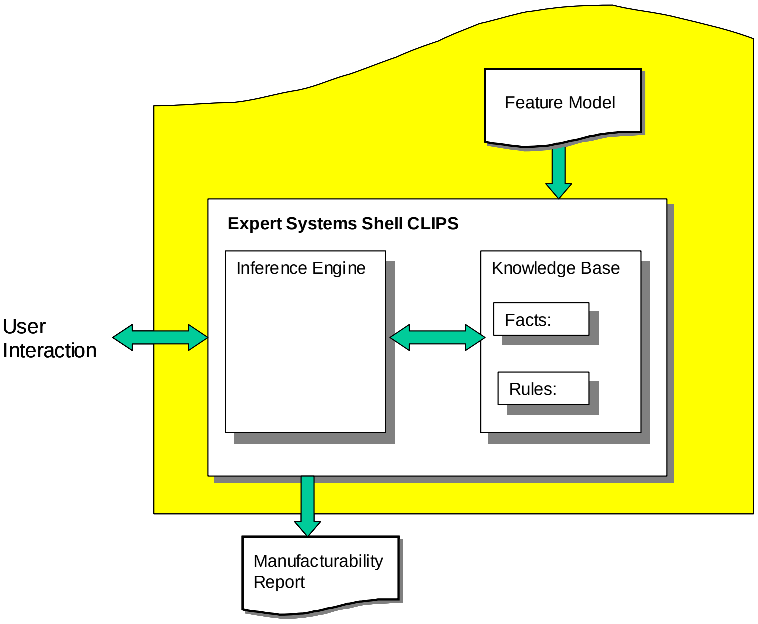


Figure 6.6 Manufacturing Advisor Expert System Architecture.

The manufacturing advisor stores manufacturing knowledge facts and rules in the knowledge base and uses the CLIPS inference engine to control program operation. Facts are data associated with the manufacturing processes and materials (such as the wall thickness, draft angle etc), and rules are the design guidelines that specify how a part should be designed to meet the requirements of a manufacturing process or material.

The top-level operation of the expert system is shown in the flow chart in Figure 6.7. The system begins by asking the user to specify the material class for their design (plastic or metallic), and the user can either specify the material class or respond that the material class is not known. If the user specifies a material the system offers the user possible manufacturing processes that are appropriate to the chosen material, and then asks the users to select a specific material appropriate to their selected process. If the material class is unknown the user is offered a list of all available manufacturing processes, and when a process is selected the system selects an appropriate material class for the process and continues. The system then enquires what detailed design information is available, and the user can specify a feature file containing feature recognition results. The system also allows the user to specify preferences for selected design parameters such as appearance or strength, which will further influence the results that are generated. Throughout the interaction the expert system's problem solving strategy is to elicit as much design information as possible to tailor the manufacturing advice that is generated.

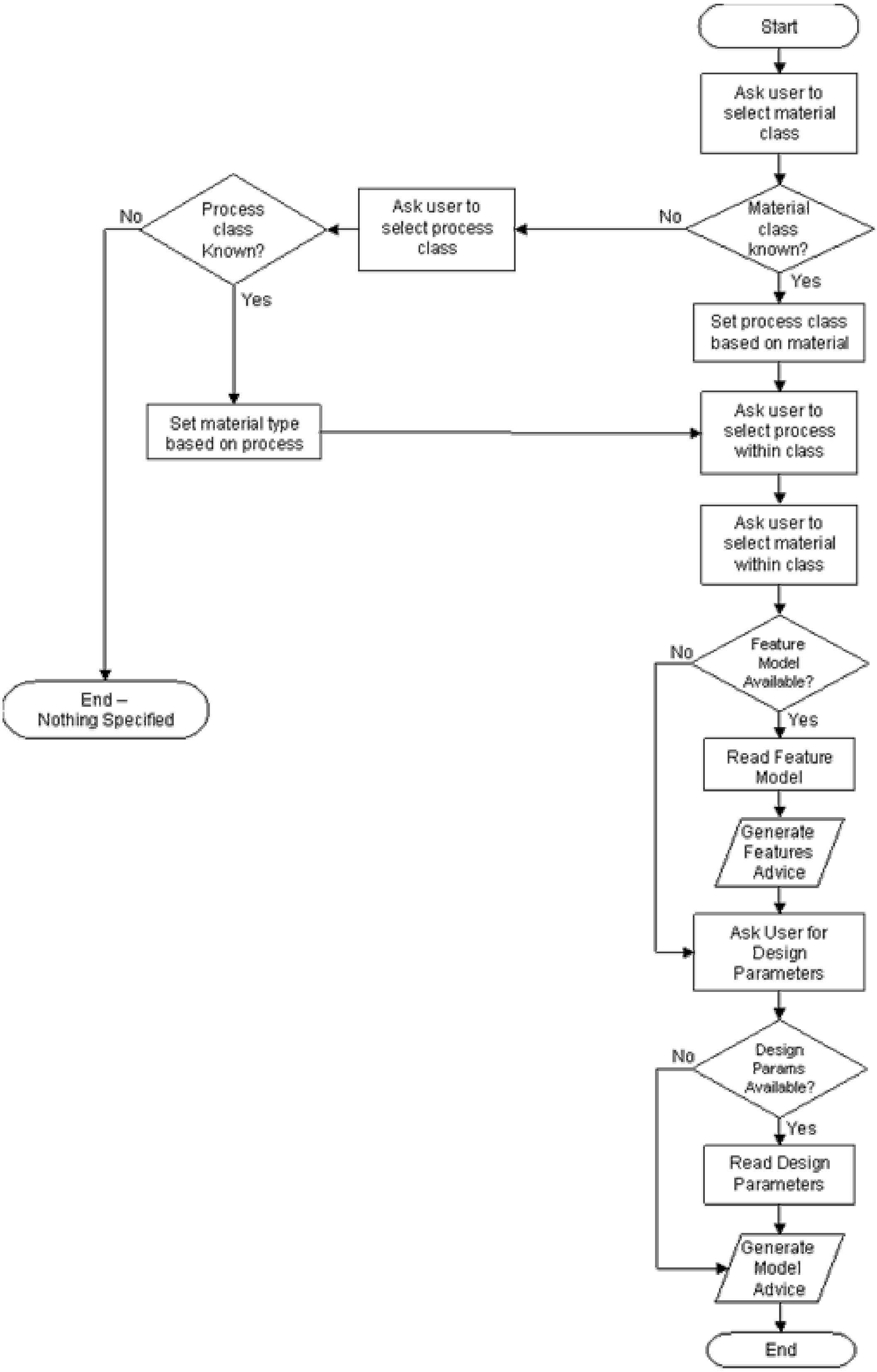

Start
Ask user to select material class
Material class known?
No
Yes
Ask user to select process class
Process class Known?
No
Yes
Set process class based on material
Set material type based on process
Ask user to select process within class
Ask user to select material within class
End – Nothing Specified
Feature Model Available?
No
Yes
Read Feature Model
Generate Features Advice
Ask User for Design Parameters
Design Params Available?
No
Yes
Read Design Parameters
Generate Model Advice
End

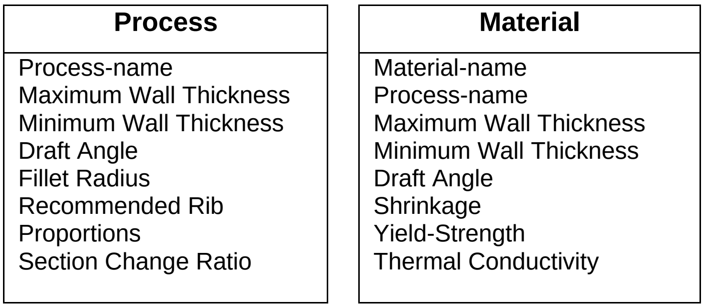
Figure 6.7 Flow Chart of Expert System Operation

The design of the knowledge base is described in more detail in the following sections and CLIPS code segments are included to illustrate the description.

6.3.4 Facts

Facts are used to store information that is known to the expert system. A fact is a basic information representation form in CLIPS, and facts can be structured using a construct called a Deftemplate that defines a set of pre-defined fields associated with the fact.

The manufacturing advisor expert system stores two types of facts:- fixed data which is the knowledge associated with particular materials and moulding processes, and modifiable data which is input into the system at run time and may be different for each session. Figure 6.8 shows the knowledge templates that have been defined to store process and material knowledge in the system. The templates can be thought of as similar to an object-oriented class and its associated attributes. The Process template stores the generic design attributes for a particular manufacturing process. The Material template stores the design attributes for a specific material and manufacturing process. Some attributes appear in both the Process and the Material template, but the Material value will override the Process value if it is available because it contains the most detailed information.



Process

Process-name
Maximum Wall Thickness
Minimum Wall Thickness
Draft Angle
Fillet Radius
Recommended Rib Proportions
Section Change Ratio

Material

Material-name
Process-name
Maximum Wall Thickness
Minimum Wall Thickness
Draft Angle
Shrinkage
Yield-Strength
Thermal Conductivity

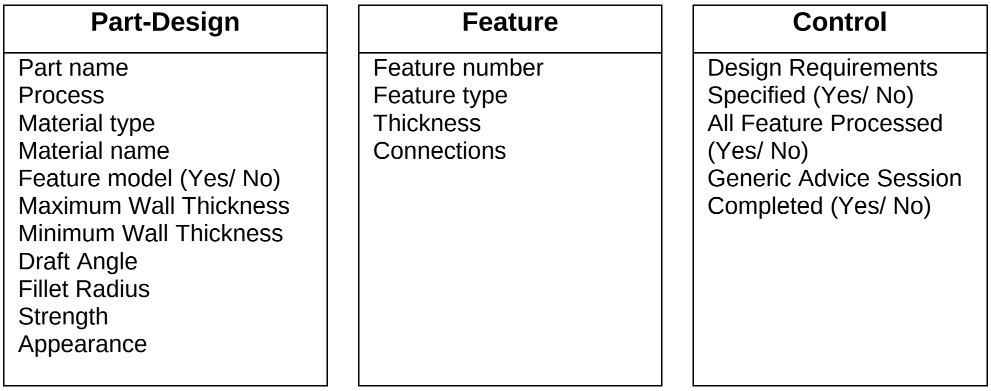
Figure 6.8 Knowledge Templates for Process and Material data

Figure 6.9 shows two sample facts written in the CLIPS syntax. The first defines the manufacturing parameters for the sand-casting process, and the second defines the manufacturing parameters for the acrylic material, and injection-moulding process.

```
(process (proc-name sand-casting)
            (max-wall-tk 127.0)
            (min-wall-tk 6.35)
            (section-change-ratio 4.0)
            (min-draft-angle 1.0)
            (typ-draft-angle 2.0)
            (rec-rib-prop 1.0)
            (rec-buttress-prop 1.0))

(material (material-name acrylic)
            (process-name injection-moulding)
            (min-wall-tk 0.6)
            (max-wall-tk 3.0))
```

Figure 6.9 Sample Facts in CLIPS Language

The modifiable facts in the knowledge base are also stored in templates. Default values are assigned to each attribute when the system is initialised, and then updated values are input into the templates as the session progresses. Two templates are used to encode the design data for the part, and a third is used to monitor and control the progress of the manufacturing advisor. Figure 6.10 shows the knowledge templates for the design data in the knowledge base.



Part-Design

Part name
Process
Material type
Material name
Feature model (Yes/ No)
Maximum Wall Thickness
Minimum Wall Thickness
Draft Angle
Fillet Radius
Strength
Appearance

Feature

Feature number
Feature type
Thickness
Connections

Control

Design Requirements Specified (Yes/ No)
All Feature Processed (Yes/ No)
Generic Advice Session Completed (Yes/ No)

Figure 6.10Knowledge Templates for Design Data

6.3.4.1 Rules

The rules are used to represent the manufacturing knowledge in the expert system and are stored as premise/ action pairs. Rules in CLIPS are defined as an antecedent (the “if” portion of the rule) and a consequent (the “then” portion of the rule). A pattern-matching operation is performed to match the antecedent conditions and determine whether the rule should be fired, and if the conditions are met then the set of actions in the consequent is executed. The manufacturing advisor expert system stores two types of rules in the system, firstly rules to elicit design information from the user or from a feature file, and secondly manufacturing rules to generate manufacturing advice. The knowledge elicitation rules form the user interface for the system, and guide the user to provide all the required information. The manufacturing rules automatically generate manufacturing advice for the design information that is input to the system. Figure 6.11 shows examples of each of the rule types.

Knowledge Elicitation Rule	**Manufacturing Rule**
If *Premises* (Material-Type = Metal) and (Material-Name = unset) are true then Perform *Actions* Print (Is the part to be manufactured from aluminium, copper, zinc or steel?) Read response Assign value to template (Part-Name (material-name matl))	If *Premises* (Process-Type = Injection Moulded) and (Strength is not important) and (wall-thickness is too thick) then Perform *Actions* Print Advice (Wall is too thick, and strength is a low consideration. You are recommended to redesign the wall with a smaller wall thickness)

Figure 6.11 Example Rules

Flexibility is achieved in the system by controlling the rule firing based on the user's responses to the knowledge elicitation rules. The user can also respond "unknown" to any question, which will trigger the system to ask further questions that will allow it to automatically assign appropriate answers.

Rules are also used to encapsulate the manufacturing guidelines that have been collated from design handbooks. An example of a manufacturing rule to check whether the section change ratio on a part is acceptable is shown in figure 6.12. The rule first checks whether all the required information is available, and that the rule has not previously been fired. If the conditions are met the rule then fires and checks the variation in wall thicknesses for the part, and writes advice to the report file.

```
;; Section change ratio
;;
;; GENR6
;;
(defrule section-change  ""
      (not (part-design (design-req unset)))
      (wall-tk-var-set ?)
      ?mypart <- (part-design (process ?proc1) (all-features-
      proc yes)(draft-angle ?angle2) (min-wall-tk ?min1) (max-
      wall-tk ?max1) (appearance ?app))
      (process (proc-name ?proc1)(section-change-ratio ?scr))

      =>
      (bind ?var  (abs (* (/ (- ?min1 ?max1) ?max1) 100)))
      (if (> ?var 10)
        then
         (printout writefile
            t crlf
            “ADVICE: The design should not have abrupt section
            changes." t crlf
            "Where a section change is required a gradual taper
            of " ?scr " must be applied (GENR6)" t crlf
              )
         )
)
```

Figure 6.12 Example rule in CLIPS language

6.3.4.2 Inputs

The knowledge base is able to accept inputs via an interactive session with the user, and from an interface file generated by the feature recognition module. The process and materials information is defined through the interactive session, and the detailed feature information is input using the interface file. The inputs that are elicited from the user through the interactive session are:

- Process Type
- Material Type
- Nominal values for maximum/ minimum wall thickness and draft angle
- User preferences for design parameters such as strength and appearance.

The interface file is generated by the feature recogniser, and an example of the file format can be seen in Figure 6.5.

6.3.4.3 Outputs

The expert system generates the manufacturing advice as a text report. The report file contains a summary of the design information that was input during the current session, and the manufacturing advice that has been generated. A sample report for a poorly designed part is shown in Figure 6.13.

```
 ************ OUTPUT FROM KNOWLEDGE BASED MOULDING ADVISOR*************
   ***************WRITTEN BY HELEN LOCKETT*************************
  **********MANUFACUTRING ADVICE FOR PART Plastic-Varying-Wall********

INFORMATION: The material has been set to acetal
INFORMATION: The process-type has been set to Injection Moulding
INFORMATION: A feature model has been read from the file plas1.txt
INFORMATION: Feature 20 is of type main wall and has thickness 4.0  mm
INFORMATION: Feature 5 is of type main wall and has thickness 4.0  mm
INFORMATION: Feature 6 is of type main wall and has thickness 4.0  mm
INFORMATION: Feature 7 is of type main wall and has thickness 4.0  mm
INFORMATION: Feature 8 is of type main wall and has thickness 4.0  mm
INFORMATION: Feature 9 is of type main wall and has thickness 4.0  mm
INFORMATION: Feature 10 is of type main wall and has thickness 4.0  mm
INFORMATION: Feature 11 is of type main wall and has thickness 4.0  mm
INFORMATION: Feature 12 is of type main wall and has thickness 1.0  mm
INFORMATION: Feature 14 is of type main wall and has thickness 4.0  mm
INFORMATION: Feature 15 is of type main wall and has thickness 4.0  mm
INFORMATION: Feature 16 is of type main wall and has thickness 4.0  mm
INFORMATION: Feature 17 is of type main wall and has thickness 4.0  mm
INFORMATION: Feature 19 is of type main wall and has thickness 4.0  mm
INFORMATION: Feature 2 is of type boss and has thickness 2.0  mm
INFORMATION: Feature 3 is of type boss and has thickness 2.0  mm
INFORMATION: Feature 4 is of type buttress and has thickness 1.4  mm
INFORMATION: Feature 13 is of type buttress and has thickness 1.4  mm
INFORMATION: Feature 18 is of type rib and has thickness 2.0  mm
INFORMATION: The importance of STRENGTH has been set to 0.0 on a scale
of 0.0 to 1.0
INFORMATION: The importance of APPREARANCE has been set to 1.0 on a
scale of 0.0 to 1.0

ADVICE: The specified max wall thickness is 4.0 which is greater than
the maximum wall thickness for a acetal part manufactured using
injection-moulding  (3.6 mm.)

ADVICE: The specified minimum wall thickness 1.0 is acceptable for
selected material acetal and process injection-moulding. (minimum
thickness for material is 0.8)

ADVICE: The specificed draft angle  0.0  degrees is less than
the minimum draft angle for a part manufactured using injection-
moulding you are recommended to reduce the increase the draft angle to
at least 0.5 degrees (GENR5)
ADVICE: Strength is a low consideration (0.0). it is recommended that
you redesign the part with a smaller wall thickness (GENR2)

ADVICE: Moulded parts should be designed with uniform wall thickness
(GENR1)

ADVICE: The design should not have abrupt section changes.
Where a section change is required a gradual taper of 3.0 must be
applied (GENR6)

ADVICE: All Corners should be generously radiussed (GENR7)

INFORMATION: Rib 18 has thickness  2.0 mm which is 80.0 percent of the
main wall thickness (2.5 mm )
```

```
ADVICE: For process injection-moulding the recommended rib thickness
is 60.0 percent of main wall thickness.  Rib 18 is too thick and
should be reduced to 1.5
```

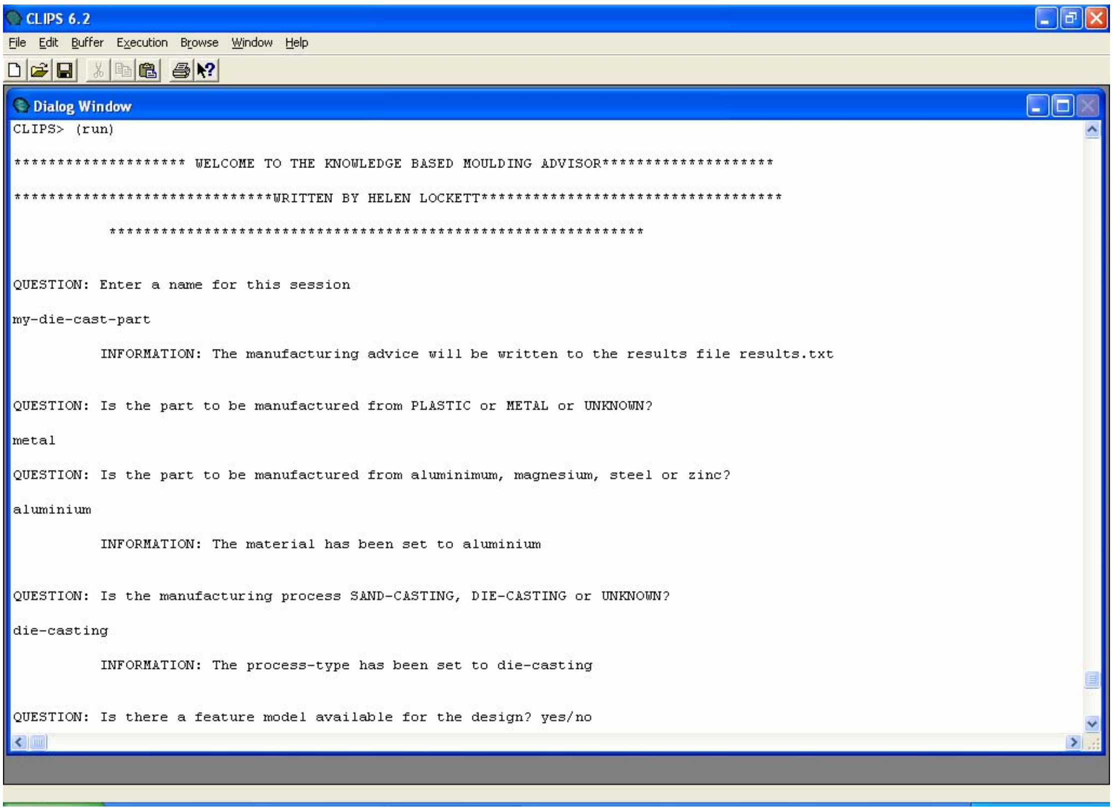
Figure 6.13 Extract from Sample output file.

6.3.5 Problem Solving Strategy

The manufacturing advisor uses the CLIPS inference engine to determine the sequence of rule firing during an advice session. Unlike a sequential program the sequence of operation in a knowledge based program is determined by the system at run time using the inference engine’s problem solving strategy. The CLIPS inference engine uses a forward chaining strategy in which the system first identifies all the candidate rules for which the antecedents are true, then uses a conflict resolution strategy to select the rule to execute, and finally executes the rule. The manufacturing advisor uses the “depth strategy” for conflict resolution, which executes newly activated rules in preference to older rules. The depth strategy is the default conflict resolution strategy in CLIPS, but other strategies are also available (CLIPS, 2005).

6.3.6 Program Interaction

The expert system is run interactively using the CLIPS environment, and reads feature information from an external file. The CLIPS user interface provides a text based dialog window to interact with the expert system, but a graphical user interface could be developed to provide a more user friendly graphical interface. Figure 6.14 shows a screen grab of an interactive session with the manufacturing advisor.



```
CLIPS> (run)

******************** WELCOME TO THE KNOWLEDGE BASED MOULDING ADVISOR********************

******************************WRITTEN BY HELEN LOCKETT***********************************

           **************************************************************


QUESTION: Enter a name for this session

my-die-cast-part

          INFORMATION: The manufacturing advice will be written to the results file results.txt

QUESTION: Is the part to be manufactured from PLASTIC or METAL or UNKNOWN?

metal

QUESTION: Is the part to be manufactured from aluminimum, magnesium, steel or zinc?

aluminium

          INFORMATION: The material has been set to aluminium

QUESTION: Is the manufacturing process SAND-CASTING, DIE-CASTING or UNKNOWN?

die-casting

          INFORMATION: The process-type has been set to die-casting

QUESTION: Is there a feature model available for the design? yes/no
```

Figure 6.14 Example Interactive Session With Knowledge Base

6.4 System Integration

The manufacturing advisor demonstrator is composed of the two main modules described above:- the feature recognition module, and the expert system module. The overall system has been integrated together using a graphical user interface developed in Visual Basic and can be run from a Windows PC. The user interface allows the user to execute the various programs in the demonstrator, view log files and visualise the results. A screen grab of the user interface is shown in Figure 6.15.

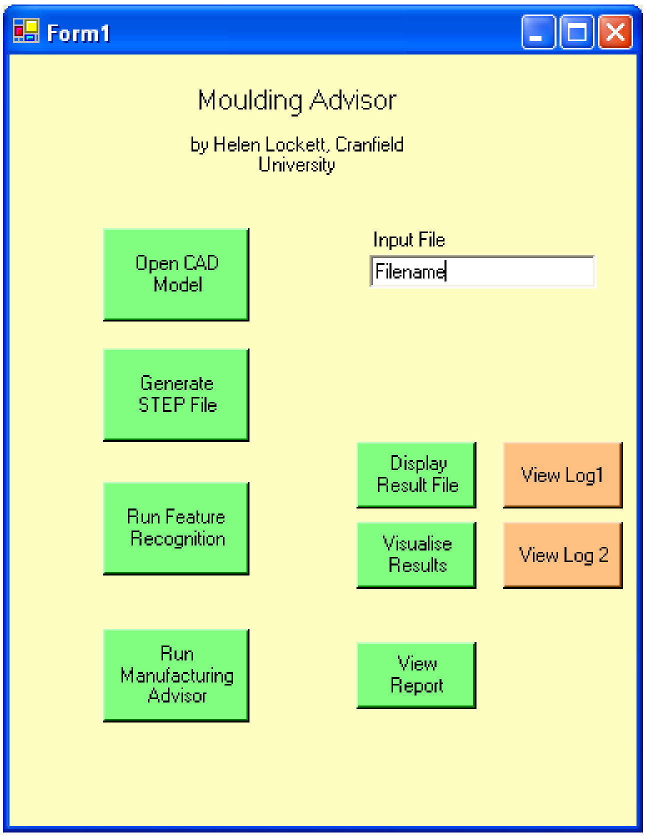


Figure 6.15 Demonstrator Graphical User Interface

6.5 Summary and Conclusions

This chapter has described the development of a demonstrator for the knowledge based manufacturing advisor and feature recognition software. The objective of the demonstrator is to provide a software environment that can be used to test the methodologies presented in this thesis.

The demonstrator has required the development of feature recognition software written in C++ and a knowledge based expert system implemented using the expert system shell CLIPS. A number of interface programs have also been written to facilitate data exchange and allow program integration.

The software development has raised a number of practical difficulties. In particular the interfacing to STEP and I-DEAS have caused some problems. The NIST STEP libraries are unsupported software, and have been found to have many limitations when used with a modern STEP translator. The I-DEAS interface has also caused some problems, particularly related to the difficulty of outputting the wall thickness values generated during the mid-surface generation process.

The demonstrator provides a core of functionality that will allow the feature recognition methodology and expert system to be tested on real world parts. The demonstrator will be used on three test cases in the following chapter.

7 TEST CASES

This chapter describes three case studies that have been analysed using the manufacturing advisor demonstrator. The first case study is the simple test part that has been used throughout the thesis. The second case study is the casing for a hi-fi remote control, and has been selected to represent a more realistic geometry including a freeform surface. The third case study is a plastic cover for a car seat adjuster and has been selected as a more challenging example to highlight the capabilities and limitations of the demonstrator. The evaluation techniques presented in Chapter 4 have been applied to each of the case studies to provide a measure of mid-surface quality and feature recognition quality for each part.

The first case study includes a more detailed description of the user interaction that is required to perform a manufacturing advice session. In the second and third case studies only the results are presented. More detailed results for case studies two and three are included in Appendix D.

7.1 Case Study 1– Simple Part

The first case study is a simple test part with some common moulding features, which has been used to demonstrate the basic principles of the feature recogniser and is based on the geometry of a sample part presented in McMahon and Browne (1998). The part will be evaluated for manufacture in aluminium using die-casting. The manufacturing advisor user interface is used to initiate the advice session, and the mid-surface model is automatically generated using the I-DEAS software. The mid-surface model has 21 faces and 57 edges. Figure 7.1 shows the CAD solid model and mid-surface abstraction for the part.

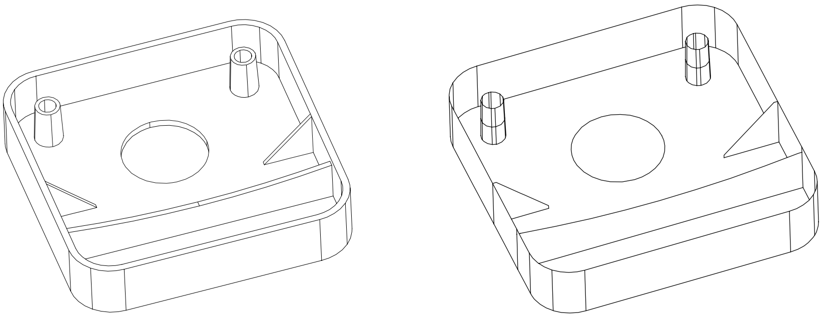
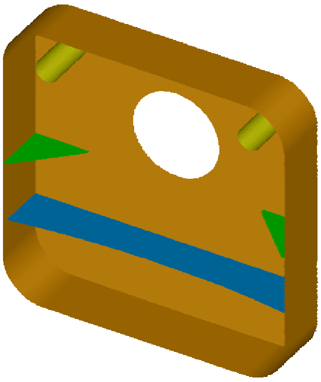


Figure 7.1 Solid Model and Mid-Surface For Simple Part

It is clear from the figure that the mid-surface model captures the important geometric characteristics of the part, including the main part wall, and attached features. A STEP AP203 file containing the mid-surface geometry is automatically generated from I-DEAS and used as input to the feature recognition software. The feature recognition results are presented to the user as a report summarising the recognised features, and a colour coded STEP file highlighting the feature types that have been identified. The feature recognition results for the simple part are shown in Figure 7.2 below.

```
SUMMARY of recognised Features
------------------------------

Loop 556 on Face 564 is a HOLE
Face 215 is a BOSS
Face 190 is a BOSS
Loop 261 on Face 263 is a BUTTRESS
Loop 601 on Face 603 is a BUTTRESS
Loop 736 on Face 738 is a RIB
```



MQF = 0.999

FRF = 1

Figure 7.2 Feature Recognition Results for Simple Part

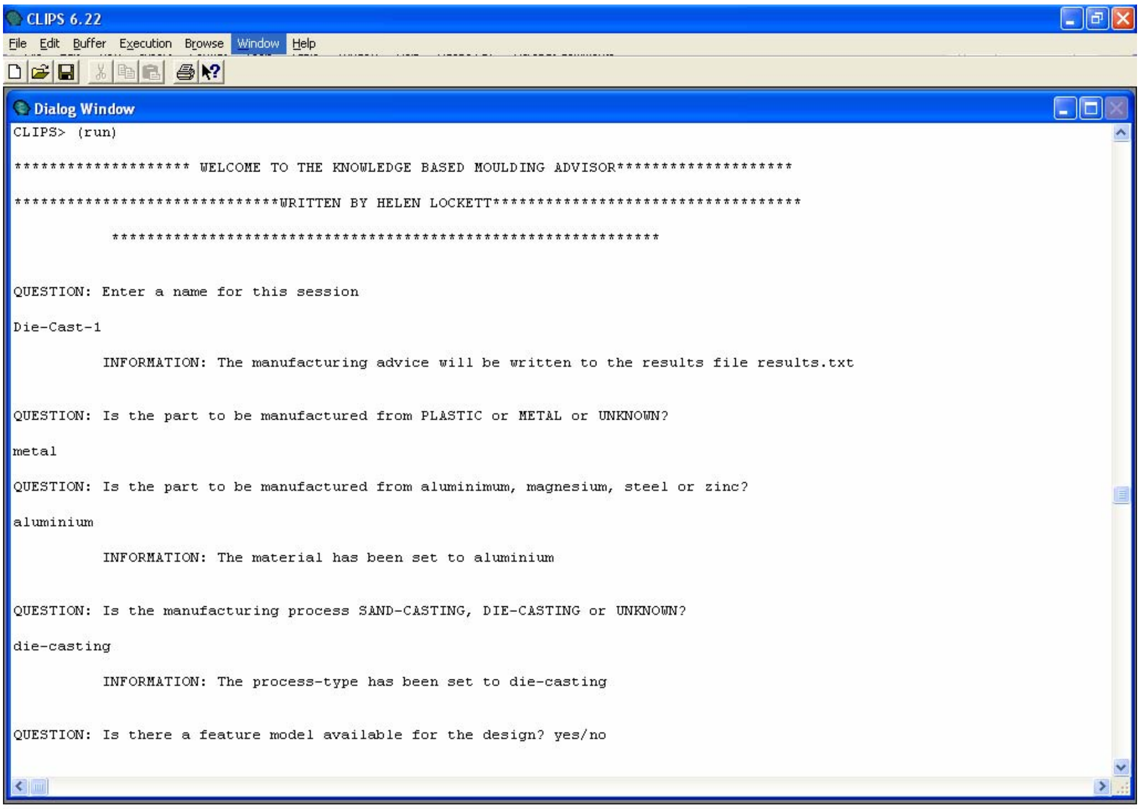
The quality of the feature recognition results for this part have already been evaluated in Chapter 4 using the two evaluation tools for mid-surface quality factor (MQF) and feature recognition factor (FRF). The evaluation results for the simple part indicate that both the mid-surface quality and feature recognition quality are very high.

The feature recognition software also generates a feature report that can be read by the manufacturing advisor expert system. The feature file contains a listing of all the recognised features, and the main attribute values for each feature. Figure 7.3 shows a section of the feature file for the part. Note that the "main–wall" features represent all the mid-surfaces that have not been identified as moulding features, and are used to compute the wall thickness variations in the manufacturing advisor.

```
(feature (feat-number 1) (feat-type hole) (has-thickness NO))
(feature  (feat-number  2)  (feat-type  boss)  (id  "F49")  (has-thickness  YES)
(thickness 2.0))
(feature  (feat-number  3)  (feat-type  boss)  (id  "F50")  (has-thickness  YES)
(thickness 2.0))
(feature (feat-number 4) (feat-type buttress) (id "F51") (has-thickness YES)
(thickness 1.4))
(feature (feat-number 5) (feat-type main-wall) (id "F42") (has-thickness YES)
(thickness 2.5))
(feature (feat-number 6) (feat-type main-wall) (id "F41") (has-thickness YES)
(thickness 2.5))
(feature (feat-number 7) (feat-type main-wall) (id "F40") (has-thickness YES)
(thickness 2.5))
(feature (feat-number 8) (feat-type main-wall) (id "F45") (has-thickness YES)
(thickness 2))
(feature (feat-number 9) (feat-type main-wall) (id "F46") (has-thickness YES)
(thickness 2.5))
(feature  (feat-number  10)  (feat-type  main-wall)  (id  "F111")  (has-thickness
YES) (thickness 2.5))
```

Figure 7.3 Feature Recognition Results Formatted for the Knowledge Base

The manufacturing advisor uses a question and answer session to obtain the initial design requirements for the part. The initial manufacturing advisor dialogue for the simple part is shown in Figure 7.4. Initially the user is asked to specify the material and process type for the part and the user is asked to input the name of the feature file. The system then asks the user to specify any additional design parameters (such as preference for strength or appearance).



```
CLIPS> (run)

******************** WELCOME TO THE KNOWLEDGE BASED MOULDING ADVISOR********************

******************************WRITTEN BY HELEN LOCKETT***********************************

           **************************************************************


QUESTION: Enter a name for this session

Die-Cast-1

          INFORMATION: The manufacturing advice will be written to the results file results.txt


QUESTION: Is the part to be manufactured from PLASTIC or METAL or UNKNOWN?

metal

QUESTION: Is the part to be manufactured from aluminimum, magnesium, steel or zinc?

aluminium

          INFORMATION: The material has been set to aluminium


QUESTION: Is the manufacturing process SAND-CASTING, DIE-CASTING or UNKNOWN?

die-casting

          INFORMATION: The process-type has been set to die-casting


QUESTION: Is there a feature model available for the design? yes/no
```

Figure 7.4 Initial Dialogue with Expert System

The manufacturing advisor writes the results of the manufacturing advice session to a text report file. The report contains a summary of the inputs that were entered into the system and the advice that is generated by the advisor. The report file for the simple part is shown in Figure 7.5.

```
*************** OUTPUT FROM KNOWLEDGE BASED MOULDING ADVISOR***************
***********************WRITTEN BY HELEN LOCKETT****************************
       *******MANUFACUTRING ADVICE FOR PART Die-Cast-1**********

INFORMATION: The material has been set to aluminium
INFORMATION: The process-type has been set to die-casting
INFORMATION: A feature model has been read from the file sc1.txt
INFORMATION: A feature model has been read from the file sc-feat.txt
INFORMATION: The nominal wall thickness has been set to 2.5
INFORMATION: Feature 2 is of type boss and has thickness 2.0 mm
INFORMATION: Feature 3 is of type boss and has thickness 2.0 mm
INFORMATION: Feature 4 is of type buttress and has thickness 1.4 mm
INFORMATION: Feature 13 is of type buttress and has thickness 1.4 mm
INFORMATION: Feature 18 is of type rib and has thickness 2.0  mm
INFORMATION: Feature 1 is of type hole
ADVICE: The specified maximum wall thickness 2.5 is acceptable for selected
material aluminium and process die-casting (maximum thickness for process is
6.35 ) (GENR2)
ADVICE: The specified minimum wall thickness 1.4 is acceptable for selected
material aluminium and process die-casting.   (minimum thickness for material
is 0.9)
ADVICE: The specified draft angle  0.0  degrees is less than the minimum draft
angle for a part manufactured using die-casting you are recommended to
increase the draft angle to at least 0.25 degrees (GENR5)
ADVICE: Moulded parts should be designed with uniform wall thickness (GENR1)
INFORMATION: The maximum main wall thickness on the part is 2.5 and the
minimum main wall thickness is 2.5
ADVICE: (Feature 3) Projections and bosses can be difficult to fill:
buttresses assist flow of such features and strengthen the component (DCR1)
ADVICE: (Feature 3) boss is of acceptable thickness
ADVICE: (Feature 2) Projections and bosses can be difficult to fill:
buttresses assist flow of such features and strengthen the component (DCR1)
ADVICE: (Feature 2) boss is of acceptable thickness
ADVICE: (Feature 18) rib. Ribs should not be square in section. Blended
sections and curves buttresses aid die filling (DCR2)
ADVICE: (Feature 18) rib has thickness  2.0 mm which is 80.0 percent of the
main wall thickness (2.5 mm )
ADVICE: For process die-casting the recommended rib thickness is 100.0 percent
of main wall thickness
ADVICE: (Feature 1) Blind holes are preferable to through holes. Through holes
can cause problems with flash. (DCR4)
ADVICE: (Feature 1) Holes should be tapered. Tapered holes assist with removal
of the casting from the die. (DCR5)
```

Figure 7.5 Manufacturing Advice Report for Simple Part

The manufacturing advisor results for the simple part identify six moulding features on the part with very high confidence. The manufacturing advisor report states that the specified wall thicknesses are acceptable for the die casting process and aluminium material. Detailed design recommendations for bosses and ribs are provided.

7.2 Case Study 2 – Remote Control Casing

The second case study has been selected as a more realistic part that is representative of a simple plastic casing of the type designed for many domestic consumer products. The solid and mid-surface models for the remote control are shown in Figure 7.6. The solid model for the remote control has 83 solid faces, and the mid-surface model has 33 faces and 99 edges.

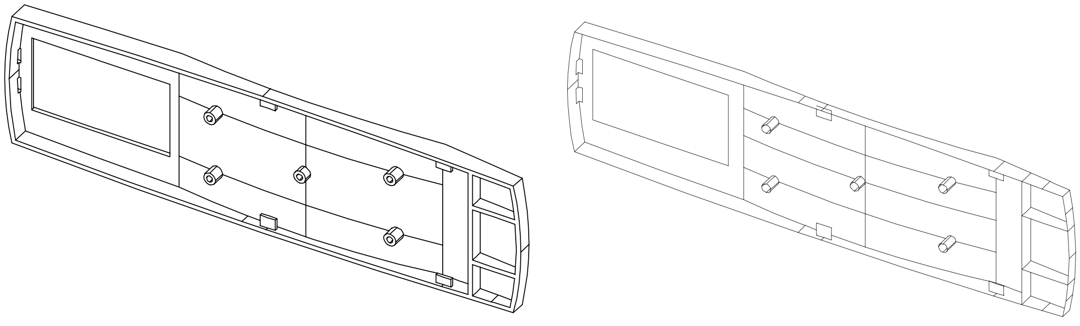
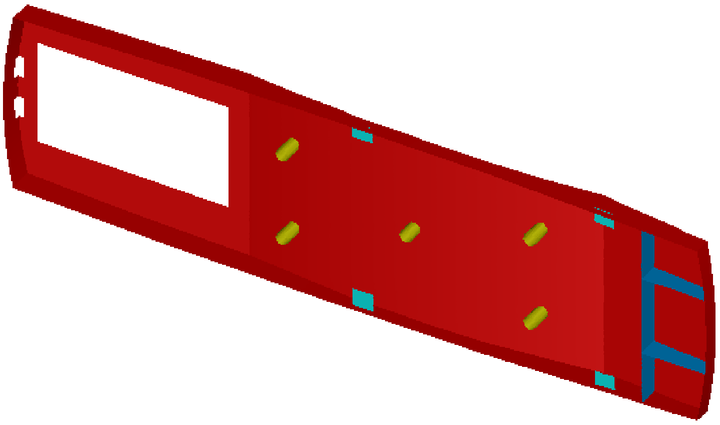


Figure 7.6 Solid and Mid-surface Models for Remote Control Part

The remote control casing presents a more challenging feature recognition problem than the simple plastic part because it has curved external surfaces, and five intersecting ribs connected to curved main wall faces. The feature recognition results for the part are shown in Figure 7.7 and show that the feature recogniser has identified one hole, five ribs, four fins, and five bosses from the part.

```
SUMMARY of recognised Features
------------------------------

Face 900 is a BOSS
Face 875 is a BOSS
Face 850 is a BOSS
Face 825 is a BOSS
Face 800 is a BOSS
Face 489 is a FIN  with non-planar parent face
Face 433 is a FIN  with non-planar parent face
Loop 1239 on Face 1241 is a HOLE
Face 377 is a FIN  with non-planar parent face
Face 321 is a FIN  with non-planar parent face
Loop 559 on Face 561 is a RIB
Loop 583 on Face 585 is a RIB
Loop 683 on Face 685 is a RIB
Loop 740 on Face 742 is a RIB
Loop 1009 on Face 1011 is a RIB
```



MQF = 1

FRF = 0.89

Figure 7.7 Feature Recognition Results for Remote Control Part

The quality of the feature recognition results has been evaluated using the MQF and FRF evaluation factors. The detailed results evaluation results are presented in Appendix D. The mid-surface quality factor (MQF) has been evaluated using a 1.5 mm spaced grid of points (equal to the main wall thickness) and a threshold value of 1.65 mm (1.5 mm +10%). The solid model and mid-surface model grids used for the evaluation contained 9341 and 5505 points respectively. The MQF is calculated to be 1.0, indicating that 100% of the solid model points are within the specified threshold distance of the mid-surface model.

The FRF for the part has been calculated to be 0.89 which indicates that some features on the part have not been recognised. The evaluation results for the simple plastic part indicate that the mid-surface quality is very high, and the feature recognition quality is high.

The FRF result of 0.89 is caused by the failure of the feature recogniser to identify the two small hole features at the end of the part. The holes have not been recognised because they are formed by the outer boundaries of adjacent faces, and not from the internal boundary of a single face. The current version of the demonstrator is only able to recognise holes that are completely inside a single mid-surface face, but this limitation could be overcome in future work.

The feature recognition results have been input to the manufacturing advisor, and an extract from the results is presented in Figure 7.8. The complete report is included in Appendix D.

```
 ************ OUTPUT FROM KNOWLEDGE BASED MOULDING ADVISOR************
    ******************WRITTEN BY HELEN LOCKETT********************
     *********MANUFACUTRING ADVICE FOR PART Remote-Control*********

      INFORMATION: The material has been set to acrylic
      INFORMATION: The process-type has been set to Injection Moulding
      INFORMATION: A feature model has been read from the file remote-
feat-file2.txt
      INFORMATION: The importance of STRENGTH has been set to 0.3 on a
scale of 0.0 to 1.0
      INFORMATION: The importance of APPREARANCE has been set to 0.9
on a scale of 0.0 to 1.0
      ADVICE: The specified maximum wall thickness 1.5 is acceptable
for selected material acrylic and process injection-moulding (maximum
thickness for process is 3.0 ) (GENR2)
      ADVICE: The specified minimum wall thickness 1.0 is acceptable
for selected material acrylic and process injection-moulding (minimum
thickness for material is 0.6)
      ADVICE: The specified draft angle  0.0  degrees is less than the
minimum draft angle for a part manufactured using injection-moulding
you are recommended to reduce the increase the draft angle to at least
0.5 degrees (GENR5)
      ADVICE: Moulded parts should be designed with uniform wall
thickness (GENR1)
      ADVICE: Variations in wall thickness should be minimised for
parts in which appearance is important. The variation in main wall
thickness is 0.0 percent which is acceptable for a part with high
importance for appearance (more than 0.5) (IMR3)
      ADVICE: The design should not have abrupt section changes. Where
a section change is required a gradual taper of 3.0 must be applied
(GENR6)
      ADVICE: All Corners should be generously radiussed (GENR7)
      INFORMATION:  Rib  27  has  thickness   1.0  mm  which  is
66.6666666666667 percent of the main wall thickness (1.5 mm )
      ADVICE:  For  process  injection-moulding  the  recommended  rib
thickness is 60.0 percent of main wall thickness.
      Rib 27 is too thick and should be reduced to 0.9
      ADVICE:  For  process  injection-moulding  the  recommended  rib
thickness is 60.0 percent of main wall thickness
```

Figure 7.8 Extract from Manufacturing Advisor Report for Remote Control

The manufacturing advisor results for the remote control part have identified fifteen moulding features with high confidence. The specified wall thicknesses are acceptable for the selected manufacturing process and material, but the rib proportions for the attached rib should be reviewed because they are too thick compared to the main wall part thickness.

7.3 Case Study 3 – Car Seat Adjuster

The third case study has been developed to demonstrate the capabilities and limitations of the manufacturing advisor when applied to a more realistic part with complicated feature interactions. The case study is presented in two parts – the first test is performed using mid-surface geometry generated automatically by I-DEAS and highlights some limitations of the mid-surface generation, and the second test is performed using the mid-surface geometry after some manual editing to improve the mid-surface quality.

The automotive industry is a major user of mass produced injection moulded plastic parts. The parts must be manufactured to engineering tolerances, and meet the functional and aesthetic requirements of the user. The part that has been used in this case study is a seat adjuster lever based on an image in (Ticona, 2000). The part geometry, and the mid-surface model are shown in Figure 7.9.

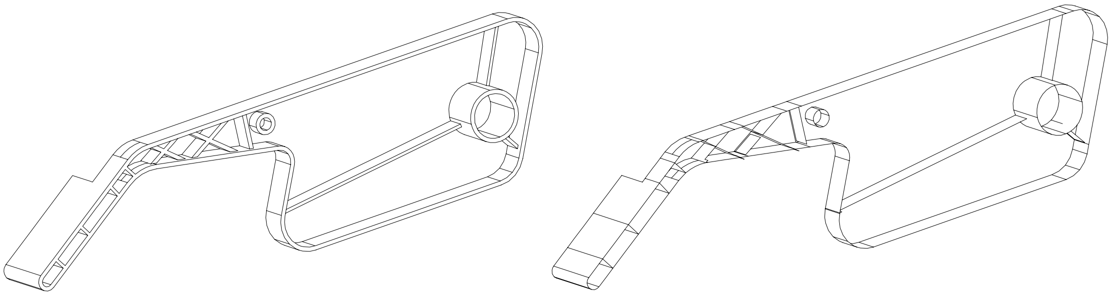


Figure 7.9 Solid Model and Mid-surface Model for Seat Adjuster Part.

The mid-surface generation for the part appears to be successful, and all the main surfaces of the part are represented in the mid-surface model. However, although the shape of the part is in most cases correctly represented, the connectivity between the faces on the mid-surface is poor in some areas, and some additional unwanted faces have been generated in the model. Figure 7.10 shows a zoomed view of some of the poor quality mid-surfaces. It can be seen in the figure that there are some unwanted faces added to the outside of the part, and some of the mid-surfaces on the inside of the part have not been correctly connected together.

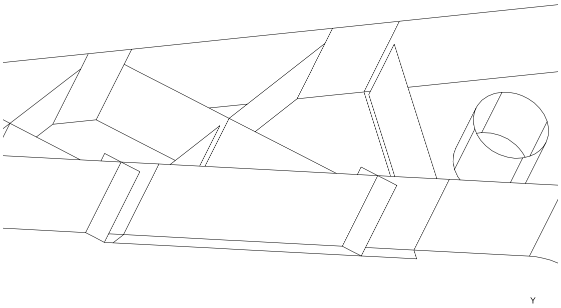
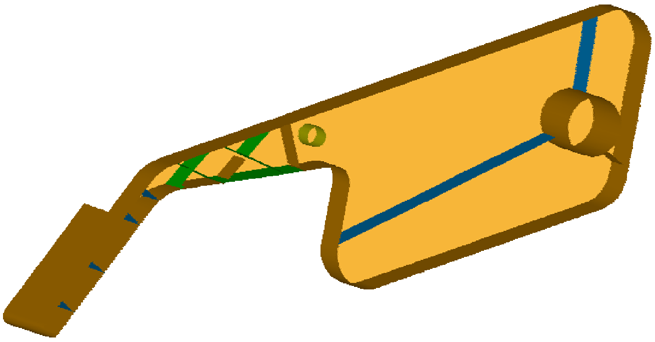


Figure 7.10 Example of Poor Quality Mid-surface Generation

The feature recognition has been performed for the seat adjuster part and the results are shown in Figure 7.11. It can be seen from the results summary that many features have been recognised, however from the results visualisation it is clear that some of the ribs have been incorrectly identified as buttresses, and others have not been recognised at all. Some main wall faces have also been identified as buttresses, due to their connectivity to the unwanted additional faces.

The evaluation techniques have been applied to the part to check the quality of the results (the detailed results are presented in Appendix D). The MQF is calculated to be 0.9998 indicating that the mid-surface quality if very high, and the FRF results is 1.24, indicating that too many features have been recognised on the part. It is clear that there are some problems with the feature recognition, and the user would be recommended to investigate the source of the problem. The feature recognition problems in this case study are cause by the poor connectivity between the mid-surfaces in the part.

```
SUMMARY of recognised Features
------------------------------
Face 390 is a BOSS
Loop 298 on Face 300 is a BUTTRESS
Loop 346 on Face 348 is a BUTTRESS
Loop 542 on Face 544 is a RIB
Loop 785 on Face 787 is a RIB
Loop 1149 on Face 1151 is a BUTTRESS
Loop 1203 on Face 1205 is a BUTTRESS
Loop 1227 on Face 1229 is a BUTTRESS
Loop 1251 on Face 1253 is a BUTTRESS
Loop 1323 on Face 1325 is a BUTTRESS
Loop 1356 on Face 1358 is a BUTTRESS
Loop 1388 on Face 1390 is a BUTTRESS
Loop 1411 on Face 1413 is a BUTTRESS
Loop 1469 on Face 1471 is a BUTTRESS
Loop 1523 on Face 1525 is a BUTTRESS
Loop 1594 on Face 1596 is a BUTTRESS
Loop 1892 on Face 1894 is a RIB
Loop 1980 on Face 1982 is a RIB
Loop 2080 on Face 2082 is a RIB
Loop 2232 on Face 2234 is a RIB
```



MQF = 0.9998

FRF = 1.23

Figure 7.11 Feature recognition Results for Unedited Seat Adjuster Part.

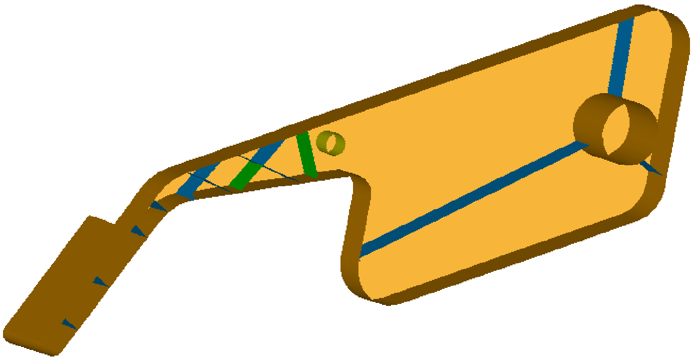
The feature recognition process for the seat adjuster part has been repeated after manually “cleaning” the mid-surface geometry. Two changes have been made to the part – firstly the unwanted extra faces have been deleted from the mid-surface model (5 faces were deleted), and secondly the badly connected mid-surface have been trimmed/ extended and stitched to improve the model connectivity (in total 3 surfaces were extended, and 9 were stitched). These changes were made by hand, although it would be possible to automate this process if required.

The feature recognition results for the “cleaned” mid-surface model are shown in Figure 7.12. The results for the cleaned model are significantly better than those for the original model. In the cleaned model all but two of the ribs have been correctly identified and none of the main wall faces have been falsely recognised as attached features. Two ribs that have been incorrectly identified as buttresses due to I-DEAS being unable to correctly connect some of the surface edges.

The results evaluation for the part shows that the confidence in the feature recognition results is now much higher (FRF = 1.04), although the value still indicates that some extra features have been incorrectly recognised. In this case the incorrect recognition of two ribs as buttresses has affected the results.

The feature recognition results for this part also highlight a limitation with the feature recognition demonstrator. The large boss on the part has not been recognised because it is connected to the outer bound of three adjacent faces, and not to the interior of a single face. This limitation could be overcome in the future by adding additional functionality to the feature recogniser to group together adjacent faces to form composite faces prior to feature recognition.

```
SUMMARY of recognised Features
------------------------------
Face 369 is a BOSS
Loop 342 on Face 344 is a BUTTRESS
Loop 521 on Face 523 is a RIB
Loop 707 on Face 709 is a RIB
Loop 788 on Face 790 is a RIB
Loop 1115 on Face 1117 is a RIB
Loop 1162 on Face 1164 is a BUTTRESS
Loop 1227 on Face 1229 is a RIB
Loop 1303 on Face 1305 is a RIB
Loop 1475 on Face 1477 is a RIB
Loop 1500 on Face 1502 is a RIB
Loop 1533 on Face 1535 is a RIB
Loop 1566 on Face 1568 is a RIB
Loop 1880 on Face 1882 is a RIB
Loop 1967 on Face 1969 is a RIB
Loop 2058 on Face 2060 is a RIB
Loop 2152 on Face 2154 is a RIB
```



MQF = 0.9998

FRF = 1.04

Figure 7.12 Feature Recognition Results for Cleaned Seat Adjuster Part

The feature recognition results have been input to the manufacturing advisor and an extract from the advice report is shown in Figure 7.13. The complete report is included in Appendix D.

```
      ********** OUTPUT FROM KNOWLEDGE BASED MOULDING ADVISOR************

   **********************WRITTEN BY HELEN LOCKETT****************************

    *************MANUFACUTRING ADVICE FOR PART Seat-Adjuster****************

INFORMATION: The material has been set to acetal
INFORMATION: The process-type has been set to Injection Moulding
INFORMATION: A feature model has been read from the file seatadj-fea-file.txt
INFORMATION: The importance of STRENGTH has been set to 1.0 on a scale of 0.0
to 1.0
INFORMATION: The importance of APPREARANCE has been set to 0.0 on a scale of
0.0 to 1.0
ADVICE: The specified maximum wall thickness 2.0 is acceptable for selected
material acetal and process injection-moulding (maximum thickness for process
is 3.6 ) (GENR2)
ADVICE: The specified minimum wall thickness 1.5 is acceptable for selected
material acetal and process injection-moulding.  (minimum thickness for
material is 0.8)
ADVICE: The specified draft angle 0.0  degrees is less than the minimum draft
angle for a part manufactured using injection-moulding you are recommended to
reduce the increase the draft angle to at least 0.5 degrees (GENR5)
ADVICE: Moulded parts should be designed with uniform wall thickness (GENR1)
INFORMATION: The maximum main wall thickness on the part is 2.0 and the
minimum main wall thickness is 2.0
ADVICE: The variation in the main-wall thickness on the part is 0.0 percent
which is acceptable for a part where appearance is not a consideration (less
than 0.5)
ADVICE: The design should not have abrupt section changes.  Where a section
change is required a gradual taper of 3.0 must be applied (GENR6)
ADVICE: All Corners should be generously radiussed (GENR7)
INFORMATION: Rib 67 has thickness 1.5 mm which is 75.0 percent of the main
wall thickness (2.0 mm )
ADVICE: For process injection-moulding the recommended rib thickness is 60.0
percent of main wall thickness
Rib 67 is too thick and should be reduced to 1.2
```

Figure 7.13 Extract from Manufacturing Advisor Report for Seat Adjuster

The manufacturing advisor results for the seat adjuster part have identified 17 moulding features with high confidence, although it was necessary to perform some manual editing on the mid-surface model to achieve these results. The specified wall thicknesses on the part are acceptable for the selected manufacturing process and material, but the rib proportions for the attached rib should be reviewed because they are too thick compared to the main wall part thickness.

7.4 Summary and Conclusions

The three case studies that have been presented in this chapter show that the manufacturing advisor demonstrator can be used to identify moulding features from designs created in a CAD solid model, and manufacturing advice can be generated for the recognised features. The demonstrator results show that the methodologies presented are applicable to a range of realistic parts.

The two evaluation tools have been used successfully to check the feature recognition results obtained from the system. Both techniques have been shown to provide useful measures for checking the feature recognition results, and have been successfully applied by hand to all the case studies. It would also be possible to integrate these techniques into the manufacturing advisor demonstrator to automatically evaluate the mid-surface and feature recognition quality within the software tool.

8 DISCUSSION

This thesis has presented a knowledge based manufacturing advisor for CAD. The primary objective of the project has been to develop techniques that will help designers to incorporate manufacturability requirements into their designs from an early stage in the design process. This objective has been met through the development of a feature recognition approach for moulded parts linked to a manufacturing advisor expert system.

A comprehensive literature review has been undertaken which has identified a number of existing manufacturing advisor tools for moulding processes. Existing tools have been found to either offer advice on moulding machine selection and costing, or to offer design advice based on a simple geometric description that is input directly to the advisor tool. There are no existing manufacturing advisor tools that generate manufacturing advice for a design that has been modelled using a CAD system. There are a number of analysis tools that can perform a manufacturability evaluation using CAD geometry, but although these tools offer good integration with a CAD system they are not able to advise on how the design should be modified to improve manufacturability.

The literature review also highlighted that there has been very limited published research in the field of feature recognition for moulded parts. A status report on feature recognition research has stated that "the field is wide open for new and valuable contributions in [die casting and injection moulding]" (Han, Pratt and Regli, 2000).

The thesis has presented two main areas of research – feature recognition for moulded parts and a manufacturing advisor expert system. The research contribution in each of these areas will be discussed in the following sections.

8.1 Feature Recognition

A feature recognition methodology has been developed for thin walled parts. The methodology is significantly different to existing feature recognition techniques, firstly because it uses a non-manifold mid-surface model as the basis for feature recognition and secondly because it is applied to feature recognition for moulded parts.

Mid-surface feature recognition has been shown to have some benefits over other feature recognition techniques for this application. Many of the features that are of interest for moulding are more directly accessible from the mid-surface geometry than they are from the solid part. For example wall junctions and stiffeners can be recognised directly without the need to perform a computationally expensive search for feature interactions. The feature recognition algorithms are also able to identify moulding features from parts that have styled free-form faces, whereas most CAD feature recognition is limited to simple prismatic parts.

The developed approach also introduces some difficulties that are not present in solid model feature recognition, due to the use of an abstraction from the CAD geometry as the basis for feature recognition. One issue is that the features are not recognised on the actual CAD geometry, so additional steps are required to link the mid-surface features to the relevant faces in the CAD solid model; and a second is that the quality of the feature recognition results is dependent on the quality of the mid-surface that is generated.

The first problem can be resolved by storing a mapping between the mid-surface faces, and the solid model faces that were used to generate them. The second problem is more fundamental because there is currently no algorithm that can automatically generate a mid-surface abstraction from any 3D geometry, and there are some geometry shapes for which a mid-surface abstraction does not accurately describe the geometry. In this project the limitations of existing mid-surface generation tools have been managed through the development techniques to evaluate mid-surface and feature recognition quality. The Hausdorff distance is able to identify regions that are missing from the mid-surface, and has been found to provide a useful measure of mid-surface quality. The feature recognition results evaluation uses a face mapping approach to identify whether all the features on the part have been recognised.

There are a number of active research projects working on mid-surface generation algorithms, and the author believes that the quality of automatic mid-surface generation tools will continue to improve in the future. The I-DEAS mid-surface function used in this research was found to be reasonably functional, although it has some limitations and it is sometimes necessary for the user to make small manual edits to the mid-surfaces after they have been generated.

A mid-surface feature recognition demonstrator has been developed and tested on a range of moulded parts. The demonstrator has been found to give good results, and the evaluation techniques provide useful measures of mid-surface and feature recognition quality. The demonstrator has been implemented for a limited range of moulding features, but the functionality could be extended to a wider range of feature types in the future.

The feature recognition methodology presented in this project has been developed as part of a manufacturing advisor tool, but there are also several other potential applications for the technique. For example mid-surface representations are widely used in finite element analysis to analyse thin walled structures, and feature recognition could be used in a finite element pre-processor to identify different structural regions on the geometry to facilitate applying loads or material properties. Mid-surface feature recognition could also be applied to other thin walled manufacturing processes such as sheet metal design.

8.2 Manufacturing Advisor Expert System

A manufacturing advisor expert system for moulded parts has been presented, and a demonstrator for the system has been implemented using the expert system shell CLIPS. The main research contribution of the expert system has been the development of a flexible knowledge structure, and the integration with the CAD feature recognition. Product design is an iterative process in which design decisions are made based on the information available at that time. The manufacturing advisor has therefore been designed to allow the user to input design information at a variety of different levels of detail, and the manufacturing advice that is generated by the system is tailored to the inputs that are provided. The system can either provide generic design information for a whole part, or use the feature recognition results to provide design advice at the level of the feature.

The integration between the feature recognition and the expert system has brought significant benefits over a standalone expert system. The feature recognition results allow the expert system to evaluate the manufacturability of each feature in the designed part, and the manufacturing advisor can therefore perform a checking role to ensure that all features in the part have been designed correctly. The tool can also be used to evaluate a design for a range of alternative manufacturing processes or materials.

The main benefit of the expert system is its ability to tailor the design information that is presented to the user directly to the design on which they are working. In most organisations design for manufacture information is available, but it is stored in many sources and the designer needs to know what information to look for and how to apply it to their current problem. Collating design information in a single repository and providing an interactive system to deliver the relevant information to the user helps to ensure that the correct design guidelines are followed.

The development of the expert system has also highlighted some difficulties with developing knowledge based applications. Encoding the manufacturing knowledge is very time consuming, and it can be difficult to convert guidelines written in English into rules that can be applied in an expert system. In order to implement a fully functional advisor tool it would be necessary to encode a large number of rules, and to perform extensive testing to ensure that the rules are complete and consistent.

8.3 Future Work

This thesis has presented a feature recognition methodology for moulded parts, integrated with a manufacturing advisor expert system. A demonstrator has been developed and tested on a number of test cases. The work to date has produced a research demonstrator that has proved that the proposed methodology can be used to implement a functional manufacturing advisor tool, but further work would need to be undertaken to extend the demonstrator to a full software implementation.

Several possible areas of future work have been identified:

- Investigation of other potential areas of application for the feature recognition methodology including finite element analysis, sheet metal design and composites design.
- Further development of the feature recognition methodology to support a wider range of feature types and consider feature proximities, possibly considering negative as well as positive features.
- Improved bi-directional integration between the feature recognition software and the CAD solid model
- Testing of the feature recognition technique using an alternative mid-surface generation function
- Automation of the evaluation techniques to form part of the software demonstrator.

9 CONCLUSIONS

This thesis has presented a knowledge based manufacturing advisor for CAD. The literature review identified that existing feature techniques are not suitable for moulded parts and that most existing manufacturing advisor tools cannot be integrated with geometry from a CAD system.

A novel feature recognition technique has been developed that is able to recognise moulding features from a mid-surface model. The feature recognition technique has been tested in a software demonstrator and found to give good results for a range of moulded parts.

The main limitation of the mid-surface feature recognition is that there is currently no mid-surface generation algorithm that can automatically generate a high quality mid-surface for any arbitrary geometry. This limitation has been managed through the development of two evaluation techniques to measure the quality of the feature recognition results.

The mid-surface quality evaluation provides a means to evaluate the similarity of the mid-surface geometry to the solid geometry, and the feature recognition results evaluation formula is used to judge the completeness of the feature recognition. The mid-surface quality evaluation is generic and can be applied to any moulded part, although the accuracy of the results obtained is limited by the size of the point set used for analysis. The feature recognition results evaluation formula has been shown to work successfully for a range of shell type moulded parts, but it has not been possible to develop a generic formula that could be applied to any moulded part.

A knowledge based expert system has been developed that is able to generate manufacturing advice for several different moulding processes and materials, and is flexible enough to generate design advice for designs with different levels of design detail. The manufacturing advisor expert system uses manufacturing guidelines from design handbooks and material suppliers to generate the manufacturing advice for the user.

The final outcome of this research is a knowledge based manufacturing advisor for moulded parts that is integrated with a CAD system. The advisor uses a novel feature recognition approach to provide a link between the geometry in a CAD system and the manufacturing advisor expert system. The expert system is able to evaluate the manufacturability of an existing design, and provide guidance on how the design could be improved. The integration of the feature recognition and expert system has brought significant benefits over a standalone expert system because it allows the manufacturing advice to be tailored to specific design features in the part.

10 REFERENCES

1. Armstrong, C. G. (1994). Modelling requirements for finite-element analysis. *Computer Aided Design.* 26, (7).
2. Aspert, N and Ebrahimi, T. (2002). MESH: Measuring errors between surfaces using the hausdorff distance. In: *Proc. Of the IEEE International Conference in Multimedia and Expo (ICME) 2002,* Lausanne, Switzerland, August 26-29 2002, 1, p. 705-708.
3. Beeley, P (2001). *Foundry Technology*. Butterworth-Heinman (2nd Edition).
4. Belludi, N. and Yip-Hoi, D. (2002) High level feature recognition using approximate and partial exact CAD models. *Proceedings of DETC'02. ASME 2002 Design Engineering Technical Conferences,* Montreal, Canada, September 29 – October 2, 2002.
5. Blum, H. (1967). A transformation for extracting new descriptors of shape. In *Models for the Perception of Speech and Visual Form*, Weinant Wathen-Dunn (ed), MIT Press, Cambridge, MA, 362 – 381.
6. Boothroyd, G; Dewhurst, P; Knight, W. (2002). *Product Design for Manufacture,* 2nd ed. Marcel Dekker Inc.
7. Bralla, J. G. (1986). *Handbook of product design for manufacturing : a practical guide to low-cost production*. McGraw-Hill.
8. Chen, Y-M. Wen, C-C. and Ho, C. T. (2003). Extraction of geometric characteristics for manufacturability assessment *Robotics and Computer-Integrated Manufacturing*, 19 (4) p. 371-385.
9. Chern, J-H, Gursoz, E. L, Prinz, F. B, Hall, M.A. (1990). Geometric Abstractions using medial axis Transform. *Manufacturing International.* Atlanta, Georgia. p.41 – 56.
10. Chin, K-S. and Wong, T. N. (1996). Knowledge-based evaluation for the conceptual design development of injection molding parts. *Engineering Applications of Artificial Intelligence*, 9 (4), p. 359-376.
11. Chu, C. N. Lee, J. M. and Kashyap, R. L. (1993). Skeleton Based Design Analysis of Near Net-Shape Products. *Annals of the CIRP* 4/1/1993.

12. Clancey, W. J. (1985). Heuristic Classification. Artificial Intelligence, 27,p. 289 – 350. Cited in Jackson, P. (1999) *Introduction to Expert Systems*, 3rd ed. Addison Wesley Longman, International Computer Science Series.

13. CLIPS (2005). Public Domain Expert System. Internet Resource: http://www.ghg.net/clips/CLIPS.html (accessed May 2005).

14. Cygwin User's Guide (2004). Internet Resource: http://cygwin.com/cygwin-ug-net/ (Accessed 9th May 2005).

15. Donaghy, R. J. Armstrong, C. G. and Price, M. A. (2000). Dimensional Reduction of Surface Models for Analysis. *Engineering with Computers*, 16, p. 24 – 35

16. Donaghy, R. J. MrCune, W. Bridgett, S. J. Armstrong, C. G. et al. (1996). Dimensional Reduction of Analysis Models. *Proceedings 5th Intl Meshing Roundtable*, Sandia National Laboratories, Pittsburgh, USA, p.307- 320. http://www.fem.qub.ac.uk/Resources/publications/publications.php (Accessed June 2005)

17. Efunda (2005). Sand Casting. Internet resource: http://www.efunda.com/processes/metal_processing/sand_casting_table.cfm (Accessed 9th March 2005).

18. El-Mehalawi, M. and Miller, R. A. (2003). A database of mechanical components based on geometric and topological similarity. Part I: representation. *Computer Aided Design*, 35, p. 83 – 94.

19. EngineersEdge (2005). Sand Casting Design Guide and Considerations. Internet resource: http://www.engineersedge.com/sand_cast.htm (accessed 9th March 2005).

20. Er, A. and Dias, R. (2000). A rule-based expert system approach to process selection for cast components. *Knowledge-Based Systems*, 13 (4), p. 225-234

21. Gadh, R., Gursoz, E. L, Hall, H. A, Prinz, F. B, Sudhalkar, A. M. (1991). Feature abstraction in a knowledge-based critique of designs. *Manufacturing Review*, ASME. 4 (2) p. 115 – 125.

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

23. GE Plastics (1997). Design Guide. Internet Resource: http://www.geplastics.com/resins/global/pdf/europe_guides/desineng.pdf (Accessed March 2005).

24. Giarrantano, J.C (2002). CLIPS User's Guide, version 6.2. Internet Resource: http://www.ghg.net/clips/download/documentation/usrguide.pdf (Accessed 9th May 2005).

25. Han, J, Pratt, M and Regli, W. C (2000). Manufacturing Feature Recognition from Solid Models: A Status Report. *IEEE Transactions on Robotics and Automation.* 16 (6), p. 782 -794.

26. Hostaform. (2000) Acetal copolymer (POM) Material Data, Ticona Ltd.

27. Huttenlocher, D. P, Klanderman, A, and Rucklidget, W. J. (1993). Comparing Images Using the Hausdorff Distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*,15 (9), p. 850 – 863.

28. Huang, Z. and Yip-Hoi, D. (2002). High level feature recognition using feature relationship graphs. *Computer Aided Design* 34 (8), p. 561-582.

29. Iyer, N, Jayanti, S, Lou K, Kalynaraman Y, Ramani K. (2005). Three-dimensional shape searching : state of the art review and future trends. *Computer Aided Design*. 37 (5), p. 509 – 530.

30. Jackson, P. (1999) *Introduction to Expert Systems*, 3rd ed. Addison Wesley Longman, International Computer Science Series.

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

32. Kim, Y. S. (1992). Recognition of form features using convex decomposition. *Computer-aided Design*, 24 (9), p461 – 476.

33. Kyprianou, L. K. (1980). Shape Classification in Computer Aided Design. PhD dissertation, Christ College, Cambridge University, July 1980 (Cited In: Han, J, Pratt, M and Reglie, W. C (2000). Manufacturing Feature Recognition from Solid Models: A Status Report. IEEE *Transactions on Robotics and Automation*. 16 (6), p. 782 -794).

34. Leventon, M. A. How to Use the Hausdorff Code. Internet Resource: http://www.ai.mit.edu/people/leventon/hausdorff/ (Accessed November 2004).

35. Li, B and Liu, J, (2002). Detail feature recognition and decomposition in solid model. *Computer-Aided Design*, 34 (5), p. 405-414.

36. Little, G. Tuttle, R. Clark, D. E. R and Corney, J. (1998). A feature complexity index. *Proceedings of the Institution of Mechanical Engineers. Part C*, 212 (5), p. 405 – 412.

37. Lockett, H. L and Guenov, M. D., (2002), An intelligent manufacturing advisor for casting and Injection-moulding based on a mid-surface approach. *Proceedings of DETC'02 ASME 2002. Design Engineering Technical Conferences and Computers and Information in Engineering Conference.* Montreal, Canada, September 29-October 2, 2002

38. Lockett, H. L & Guenov, M. D., (2005), Graph-based feature recognition for injection moulding based on a mid-surface approach. *Computer-Aided Design*. 37 (2), p. 251 – 262.

39. Lu, S. C. Rebello, A. B. Miller, R. A. and Kinzel, G. L. (1995). A Volume-based Geometric reasoning approach for diecastability evaluation. *Proceedings of the 15th Computers in Engineering Conference,* Boston MA, p. 65-74

40. McMahon C, Browne, J. (1998). CAD/CAM Principles, Practice and Manufacturing Management. Addison-Wesley.

41. Meystel, A. M and Albus, J. S. (2002). *Intelligent Systems: Architecture, Design and Control.* John Wiley and Sons.

42. National Institute of Standards and Technology (NIST) (2002) - STEP Class Libraries. Internet Resource: http://www.nist.gov/scl (Accessed March 2005).

43. Newell, A and Simon, H. (1963). GPS: A program that simulates human thought, Computers and Thought, E. A. Feigenbaum and J Feldman, eds. MacGraw-Hill, New York, 1963, p.279 – 296. Cited in: Meystel, A. M and Albus, J. S. (2002). *Intelligent Systems: Architecture, Design and Control.* John Wiley and Sons. P. 5.

44. Ogniewicz, R L . (1996). Skeletonization Software (2D). Internet Resource: http://www.cs.sunysb.edu/~algorith/implement/skeleton/implement.shtml (Accessed 19th April 2005).

45. Pao, W. K. S, Ransing, R. S, Lewis, R. W, and Lin, C. (2004) A medial-axes-based interpolation method for solidification simulation. *Finite Elements in Analysis and Design*, 40 (5-6), P. 577-593.

46. Preddy, K. L; Knight, B; Cowell, D. F; Mileman, T. J. (1997). CAST-AID: a Complementary Partner to Numerical Modelling of Casting Simulation. Internet Resource: http://www.greenwich.ac.uk/~mt04/casting/castingcon97/castcon97.html (accessed 2003)

47. Ravi, B; Srinivasan, M.N. (1989) Hot spot detection and modulus computation by computer graphics. *56th World Foundry Congress.* p 12.1 - 12.8.

48. Radhakrishnan, R; Amsalu,A; Kamran, M; Nnaji, B. (1996) Design Rule Checker for Sheet Metal Components using medial axis transformation and geometric reasoning. *Journal of Manufacturing Systems.* 15 (3).

49. Rezayat, M. (1996) Midsurface abstraction from 3D solid models: general theory and applications. *Computer Aided Design*. 26 (11), p. 905 – 915.

50. Rodriguez-Toro, C; Tate, S; Jared, G; Swift, K. (2002) Shaping the complexity of a design. *Proceedings of the IMECE2002. ASME International Mechanical Engineering Congress and Exposition.* November 17 – 22, 2002, New Orleans, Louisiana.

51. Shah, J. and Mantyla, M. *Parametric and Feature Based CAD/CAM*. John Wiley, 1995.

52. Sheehy, D. J, Armstrong, C. G., Robinson, D. J. Shape Description By Medial Surface Construction. (1996) *IEEE Transactions on Visualisation and Computer Graphics* 2 (1).

53. Su, Y, Lee, K. H, Senthil Kumar, A. (2004). Automatic mesh generation and modification techniques for mixed quadrilateral and hexahedral element meshes of non-manifold models. *Computer-Aided Design*, 36 (7), p. 581-594

54. Ticona, *Designing with Plastics, The Fundamentals*. Ticona Ltd (2000).

55. Tolga Bozdana, A and Eyercioğu, O. (2002) Development of an expert system for the determination of injection moulding parameters of thermoplastic materials: EX-PIMM. *Journal of Materials Processing Technology*, 12 (1-3), p. 113-122.

56. TranscenData (1995), Medial Object Toolkit. Internet Resource: http://www.transcendata.com/TranscenData_MO_paper/TranscenData_MO_paper.htm (Accessed June 2005)

57. UGS I-DEAS NX. (2005). WWW Document: http://www.ugs.com/products/nx/ideas/ (accessed 19th April 2005)

58. Vandenbrande, J. H and Requicha, A. A. G. (1993). Spatial Reasoning for the Automatic Recognition of Machinable Features in Solid Models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15 (12), p. 1269 – 1285.

59. Vergeest, J.S.M, Spanjaard S, Song Y. (2003) Directed mean Hausdorff distance of parameterized freeform shapes in 3D – A case study. Internet Resource: http://dutoce.io.tudelft.nl/~jouke/docdb/docs/viscomp2003vergeest.pdf (Accessed 29th October 2004).

60. Wozny, M. J, Pratt, M. J; Poli, C. (1994). *Topics in Feature-Based Design and Manufacturing. Advances in Feature Based Manufacturing.* Ed: J.J. Shah, M. Mantyla & D. S. Nau. 1994. Elsevier Science.

61. Yang, S. Y; Girivasan, V; Singh, N. R; Tansel, I. N; Kropas-Hughes, C. V. (2003) Selection of optimal material and operating conditions in composite manufacturing. Part II: complexity, representation of characteristics and decision making. *International Journal of Machine Tools & Manufacture*. 43 p. 175 – 184

62. Yin, Z-P, Ding, H, Li, H-X and Xiong, Y-L. (2004) Geometric mouldability analysis by geometric reasoning and fuzzy decision making. *Computer-Aided Design*, 36 (1), p. 37-50.

63. Yin, Z-P, Ding, H and Xiong, Y-L. (2001) Virtual prototyping of mould design: geometric mouldability analysis for near-net-shape manufactured parts by feature recognition and geometric reasoning. *Computer Aided Design*, 33 (2), p 137 – 154.

64. Zhang, X, Wang, J , Yamazaki, K and Mori, M (2004) A surface based approach to recognition of geometric features for quality freeform surface machining, *Computer-Aided Design*, 36 (8) p. 735-744.

65. Zhou-Ping Yin, Han Ding, Han-Xiong Li and You-Lun Xiong (2004), Geometric mouldability analysis by geometric reasoning and fuzzy decision making, *Computer-Aided Design*, 36 (1) p. 37-50.

APPENDICES

APPENDIX A - Author Publication:

Lockett, H. L & Guenov, M. D., (2005), Graph-based feature recognition for injection moulding based on a mid-surface approach. Computer-Aided Design. 37 (2), p. 251 – 262.

Computer-Aided Design 37 (2005) 251–262

COMPUTER-AIDED DESIGN

www.elsevier.com/locate/cad

Graph-based feature recognition for injection moulding based on a mid-surface approach

Helen L. Lockett*, Marin D. Guenov

School of Engineering, Cranfield University, Cranfield, Bedford, MK 43 0AL, UK

Received 10 October 2003; received in revised form 4 June 2004; accepted 7 June 2004

Abstract

This paper presents a novel CAD feature recognition approach for thin-walled injection moulded and cast parts in which moulding features are recognised from a mid-surface abstraction of the part geometry. The motivation for the research has been to develop techniques to help designers of moulded parts to incorporate manufacturing considerations into their designs early in the design process. The main contribution of the research has been the development of an attributed mid-surface adjacency graph to represent the mid-surface topology and geometry, and a feature recognition methodology for moulding features. The conclusion of the research is that the mid-surface representation provides a better basis for feature recognition for moulded parts than a B-REP solid model. A demonstrator that is able to identify ribs, buttresses, bosses, holes and wall junctions has been developed using C++, with data exchange to the CAD system implemented using ISO 10303 STEP. The demonstrator uses a commercial algorithm (I-DEAS) to create the mid-surface representation, but the feature recognition approach is generic and could be applied to any mid-surface abstraction. The software has been tested on a range of simple moulded parts and found to give good results.

Keywords: Feature Recognition; Mid-surface; STEP; Injection-moulding

1. Introduction

1.1. Motivation

The motivation for this research has been to help product designers to incorporate manufacturing considerations into their designs early in the design process. In the moulding industry it is common for product design and manufacture to be undertaken in separate companies and manufacturability is often not considered during the initial design phase, which often causes additional design iterations to achieve a manufacturable design.

Moulding design guidelines are available in design handbooks, and specify relatively simple geometric considerations to ensure that the part will fill and cool correctly. By following these basic guidelines the designer can achieve a more manufacturable design at the initial design phase, and should be able to reduce the number of iterations to optimise the design for manufacture.

The objective of this research has been to develop a feature recognition approach for moulded parts that could be implemented as part of a manufacturing advisor to identify features that may be expensive or difficult to manufacture early in the design process. It is intended that the feature recogniser would be used at a preliminary design stage to identify the important design features for moulding evaluation.

1.2. Research approach

Feature recognition techniques for CAD have been widely researched in recent years, with most of the effort being focussed on feature recognition for machining applications. Feature recognition tools are important for the future development of Computer Aided Design because they provide better support for design for function or manufacturability than is possible in current systems.

* Corresponding author. Tel.: +44-1234-750111.
E-mail address: h.lockett@cranfield.ac.uk (H.L. Lockett).

0010-4485//$ - see front matter © 2004 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cad.2004.06.010

However, feature recognition has proved to be problematic to implement because it is difficult to develop a robust system that can be applied to a wide range of geometry types. One of the major problems in feature recognition is the difficulty of recognising feature interactions where several simple features interact to form a more complicated feature [1].

The objective of this research has been to develop a novel feature recognition approach for thin walled parts, to support design for manufacture for injection moulding and casting processes. It has been observed in the literature that there has been very limited previous research into feature recognition for these types of parts, and existing feature recognition techniques are not well suited to moulded parts. The technique that has been developed uses a mid-surface abstraction from a CAD solid model of the part as the basis for feature recognition, and recognises shape characteristics that are important for mouldability evaluation. Design for manufacturability is particularly important for moulded parts because the constraints of the manufacturing process must be taken into account by the product designer in order to achieve a manufacturable design.

The paper is organised as follows: Section 2 of the paper is a review of the relevant literature, Section 3 presents the graph structure that is used to capture the design topology and geometry, and Section 4 presents the feature recognition methodology. The implementation of a demonstrator for the methodology is presented in Section 5, with two case studies, and finally in Section 6 the advantages and limitations of the approach are discussed, conclusions are drawn and future work defined.

2. Literature review

Design for manufacture for moulded parts is particularly important because the cost and quality of parts manufactured using these processes is highly geometry dependent. Furthermore, for these mass production processes a small change in manufacturing cost can have a major effect on the economics of a particular design.

A number of research projects have developed design for moulding tools using analytical methods or design by features approaches, but there does not appear to be significant research focussed on feature recognition for moulded parts. Knight et al. [2] developed a design for casting system aimed at the manufacturing engineer in which the user builds a simplified representation of the part using a library of standard shapes (for example filleted L sections, T and cross junctions, bars wedges and plates). The manufacturability of the part is then evaluated by combining the known behaviour of the simple shapes. Rosen et al. [3] developed a design for manufacturability tool for thin walled mechanical components based on a non-manifold geometry model that allowed them to evaluate tooling costs for injection moulding and die-casting. Their research used a design by features approach, and they did not attempt to integrate their tools with a standard CAD solid modeller.

Lu et al. [4] used a voxel based approach to identify the geometry characteristics of a part (e.g. thick areas for hot-spot detection) for casting evaluation. In their approach a three-dimensional model of the part is subdivided into voxels (small cuboid volumes) and then the distance from every voxel to the part boundary is measured. Using this approach they are able to find variations in material thickness and identify potential hot spots in the casting. They assert that this approach is much more flexible than standard feature recognition techniques. Ravi and Srinivasan [5] also presented a novel method for the identification of hot spots in complex three-dimensional castings based on analysing the geometric shape of the part. In their research two-dimensional slices were taken through the part in orthogonal directions and the variations in thickness across each slice are plotted and combined to find heavy areas.

There has been a great deal of research into feature recognition for machined parts and graph based methods have been commonly used for feature recognition from CAD solid models. In graph based feature recognition the topology of the solid model is captured in a face-edge graph, and the graph is parsed to recognise high level design features. Joshi and Chang [6] developed a graph-based approach to feature recognition of machined features from a three-dimensional solid model. Their approach uses an attributed adjacency graph (AAG) to represent the part shape, where the adjacency matrix incorporates an attribute to specify whether an edge is concave or convex. Features are recognised by traversing the AAG and searching for sub-graphs corresponding to predefined features.

More recently Gao and Shah [7] have further developed this approach to the concept of an extended attributed adjacency graph that supports several face and edge attributes. Their graph representation carries several geometric attributes associated with the nodes and arcs of the graph, and offers an improved ability to recognise interacting features. Both these techniques are applicable to manifold B-rep solid models.

Mid-surfaces have been investigated extensively for their ability to simplify geometry models for finite element analysis, and they have been applied to a lesser extent to casting and moulding research. Donaghy et al. [8], and Sheehy et al. [9] use the medial axis transform for dimensional reduction in finite element modelling. They search for geometry entities in the model that can be simplified by reducing their dimensionality; for example a long slender face may be reduced to a beam. Their work focuses on simplifying the part geometry through dimensional reduction, but it does not attempt to recognise design or manufacturing features from the geometry model. One technique that can be used to create a mid-surface from a solid model is the medial-surface transform.

The medial-surface transform is the three-dimensional equivalent of the medial-axis transform that was first proposed by Blum [10]. Algorithms that can calculate the medial-axis transform are now well developed, but a robust algorithm that can calculate the medial surface of any three-dimensional object is still the subject of research.

Several commercial tools offer working implementations of medial-surface or mid-surface algorithms. The medial surface transform from FEGS Ltd calculates the locus of an inscribed maximal sphere as it rolls around the interior of the part [11]. The mid-surface function implemented in EDS I-DEAS NX Series [12] selects pairs of faces from the solid part and calculates the mid-surface between each face pair; it then uses surface extension and trim operations to generate a mid-surface of the complete part.

3. Methodology

In this project a novel feature recognition approach for thin walled moulded parts has been developed which allows features to be recognised from the mid-surface abstraction of a 3D CAD solid model. It has been observed from the literature that the majority of previous feature recognition research has focussed on recognising machining features from fairly simple prismatic shaped parts, and follows a subtractive approach that reflects the material removal in a physical machining process. Feature recognition for moulded parts requires a significantly different approach because moulded parts often have complex freeform faces, and the manufacturing process is not sequential as it is for machining. The feature recognition methodology that has been developed in this research uses a mid-surface abstraction from the part geometry as the basis for feature recognition, and the feature recogniser parses a mid-surface geometry graph to identify moulding features. Using an abstraction from the CAD geometry as the basis for feature recognition means that the features that are recognised are not directly associated with the original CAD geometry; however, associativity to the CAD geometry can be maintained by storing the surface pairing information that is generated during the mid-surface creation. It should be noted that this solution is not generic, and would be dependent on the algorithm that had been used to generate the mid-surface geometry.

3.1. Moulding features

Moulding processes make significant demands on the product designer because, despite their flexibility to manufacture complex shapes, they also require that parts be designed with regard to the constraints of the manufacturing process. A review of design handbooks has shown that the features of interest for moulding evaluation are wall intersections, protuberances and wall thicknesses on the designed part [13,14].

For plastic parts any increase in wall thickness will significantly increase the cooling time of each part that is manufactured, and thick walls may affect the part quality. The designer therefore needs to make use of ribs and other features to provide strength and rigidity on the part without increasing the wall thickness. The intersections between stiffeners and external walls need to be designed to allow the plastic to flow easily between walls, and to cool evenly without causing warping or sinking.

3.2. Mid-surface representation

The mid-surface of a part is a dimensionally reduced representation in which the part walls are modelled as surfaces with zero thickness. It can be visualised as the locus of the centre of a maximal sphere rolling around the interior of the part. For thin walled parts the mid-surface representation offers a simplified geometry and topology representation, which retains the main design characteristics of the original part [9]. Mid-surface representations have been widely used in engineering analysis for thin walled moulded parts. For example in finite element analysis it is common to use a mid-surface abstraction of a thin walled part to reduce the computational cost of performing the finite element analysis and for flow analysis in injection moulding several simulation tools use a mid-surface abstraction for mould filling simulations [8].

A mid-surface representation is able to accurately model the shape of cast and injection moulded parts because these manufacturing processes require that parts be designed with thin, and relatively constant wall thickness. The mid-surface representation has a significant benefit over the true CAD geometry for moulded parts because the wall intersections that are important for mouldability analysis are directly accessible from the mid-surface without the need to process complex feature interactions.

In this research the EDS I-DEAS NX mid-surfacing function has been selected because of its availability and relatively robust functionality. However, mid-surfacing tools in other CAE systems (Pro/Engineer, MSC/PATRAN and others) can produce similar results. The quality of the mid-surface that can be generated by this and other mid-surface algorithms is largely dependent on the shape characteristics of the geometry that is presented to them.

An evaluation of the use of mid-surfaces to represent the shape of moulded parts has been undertaken and published separately as part of the development of a knowledge based manufacturing advisor for CAD [15]. The I-DEAS mid-surface function was evaluated and found to be able to automatically generate accurate mid-surfaces for a range of parts, but in order to achieve good results it was important for the wall thickness to be small relative to the size of the features on the part and to be relatively constant. Small features and fillets may be lost during mid-surface generation, but these small features are of less importance for manufacturability evaluation than the overall layout and proportions of walls and junctions.

3.3. Data exchange

In order to achieve a generic basis for the feature recognition process the data exchange with the CAD system has been implemented using a STEP AP203 physical file. The mid-surface representation is essentially a collection of faces that are connected along shared edges. The mid-surface is a non-manifold geometry because it contains edges that are adjacent to two, three or more faces but it is only a limited subset of a non-manifold geometry because it contains only faces bounded by edges with no dangling edges or closed volumes.

The mid-surface geometry has been exported from I-DEAS using the AP203 Manifold Surfaces with Topology representation, which although it is a manifold surface representation, does provide some support for non-manifold geometries through its support for multiple open_shell entities in a model. There is some ambiguity in the implementation of the STEP AP203 standard by different software vendors and in the I-DEAS implementation the concept of multiply connected faces is fully supported in the AP203 output.

3.4. Geometry representation

Graph based methods have commonly been used for feature recognition from CAD solid models for machining applications [6,7]. In a graph based approach a separate data structure is built to represent the geometry and facilitate the search for features. Features are recognised from the geometry graph by parsing the graph or matching sub-graphs to predefined templates. For example in the face adjacency graph (FAG) described by Shah and Mantyla [1] the nodes of the graph represent the faces of the object, and the arcs represent the connectivity (edges) between those faces. The FAG is suitable for representing the topology of manifold solid models, but it is not suitable for representing non-manifold geometries because the formulation of the graph requires that each edge must connect exactly two faces, which is not true for non-manifold geometries such as the mid-surface.

An example is shown in Fig. 1 where a simple T-shaped part has been modelled as a B-rep solid model. The model has 10 faces and 24 edges and the associated FAG graph has 10 nodes representing the faces of the part, and 24 arcs representing the edges connecting the faces. The feature recognition techniques that have been developed for this type of geometry graph classify edges as concave, convex or smooth, and match patterns of concave/convex edges to recognise features [1]. This approach is suited to prismatic parts, but the features that are of interest for moulding evaluation are difficult to identify by this means. For example to recognise the T-shaped feature in Fig. 1 the interaction between two adjacent 'step' features and the base part would need to be recognised.

In a mid-surface model the faces may be connected in any combination, without the limitations defined for a manifold solid model. The mid-surface model is subset of a non-manifold geometry in which two-dimensional entities (faces) are bounded by edges, and may be multiply connected to other faces along their edges. A mid-surface adjacency graph (MAG) has been developed to represent the mid-surface model topology for feature recognition. The standard FAG cannot be used for the mid-surface model because an edge can be adjacent to two, three or more faces, and this relationship cannot be represented on an FAG. The MAG therefore represents both the faces and edges of the model as nodes on the graph, with the graph arcs representing the connectivity between those faces and edges.

Fig. 2(a) shows the mid-surface representation for the T-shaped part, and contains three faces representing the three walls in the part, and 10 edges bounding those faces. The MAG shown in Fig. 2(b) has 13 nodes representing the geometry entities in the model (three faces, and 10 edges), and 12 arcs representing the connectivity between the entities. By observation from the MAG shown in Fig. 2(b) it is clear that edge E1 represents a junction between three

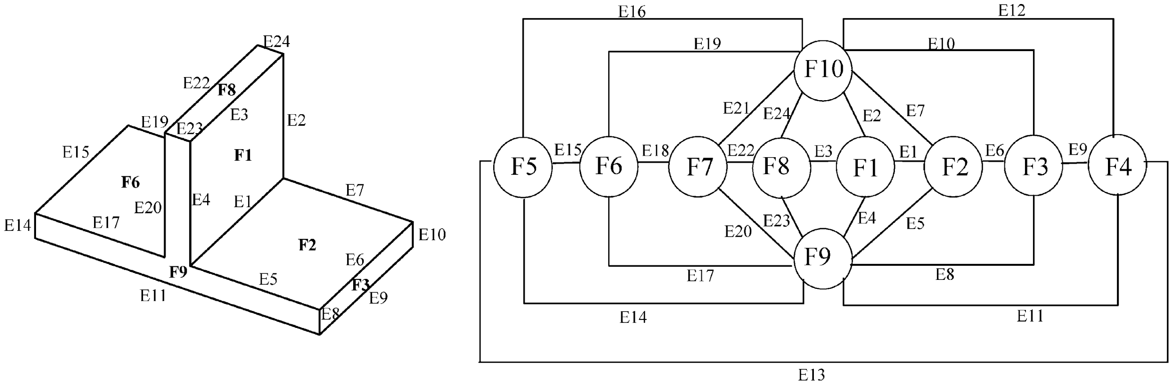


Fig. 1. Face adjacency graph for a simple T-shaped part.

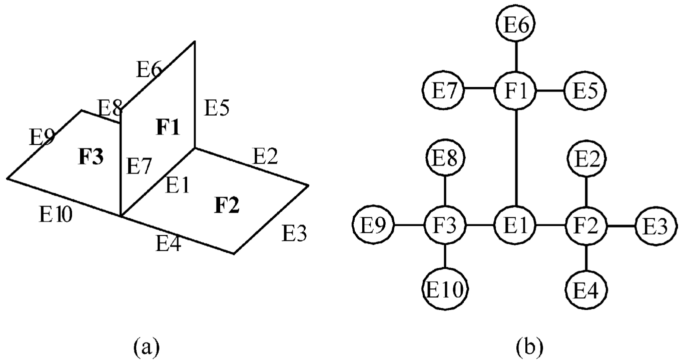


Fig. 2. Mid-surface representation and its associated mid-surface adjacency graph.

faces because it is connected to faces F1, F2,and F3. This property provides much more direct access to wall relationships than is available from the original CAD geometry and the FAG.

The MAG alone does not contain sufficient information to fully represent the mid-surface topology so it has been augmented to additionally capture important geometric attributes. An attributed MAG (AMAG) has therefore been developed that additionally identifies whether edges that belong to internal or external edge loops, and identifies edges that are reused in edge loops (for example where an edge extends to the interior of face).

An object oriented mid-surface model has been developed to store the mid-surface geometry and topology. The mid-surface model is an intermediate data structure that captures the mid-surface information generated by the CAD system, and is used to generate the AMAG. The mid-surface model stores the topology of the imported mid-surfaces along with selected geometric attributes such as face normal directions, curve characteristics, and surface types. The object model for the mid-surface model is shown in Fig. 3 below.

4. Feature recognition process

The feature recogniser that has been developed parses the AMAG described in Section 3 and searches for known patterns that identify moulding features. A face-edge adjacency matrix is constructed to represent the AMAG, and an algorithmic approach is used to perform the feature recognition. Further geometric evaluation is performed using the mid-surface model.

4.1. Feature classification

For manufacturability evaluation it is necessary to recognise high-level design features that will impact the manufacturability of the part. The types of features that are important for moulded parts are ribs, bosses, buttresses, and holes as well as feature relationships such as the junctions between walls. In moulding processes these features considered with the wall thickness have a major impact on the manufacturability of the part, and its cost and quality.

Each face in the mid-surface model represents a wall or wall segment in the part, and can be considered to be a manufacturing feature. The feature recognition problem is

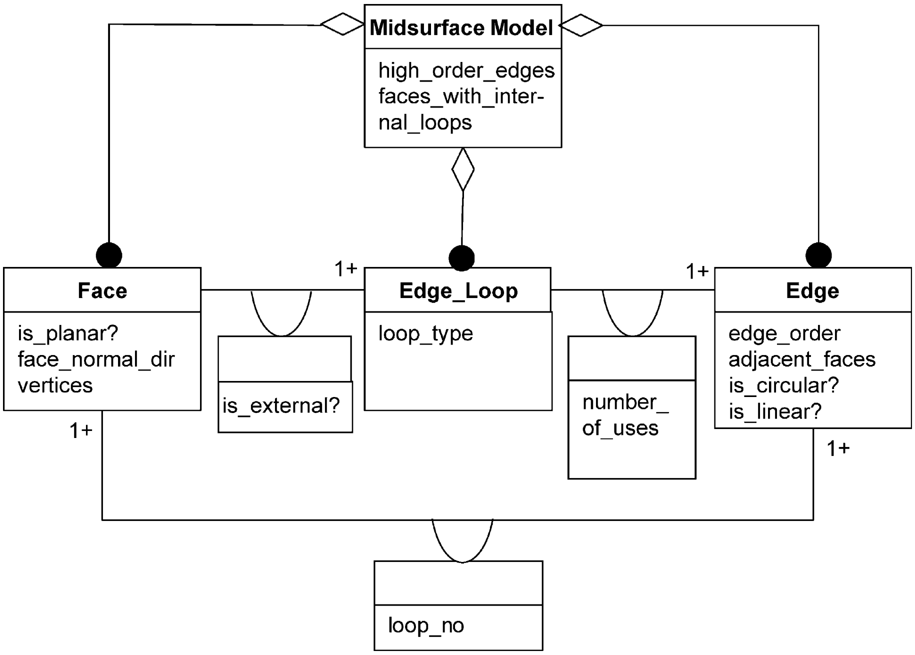
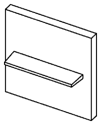
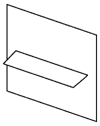
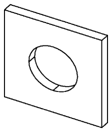
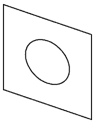
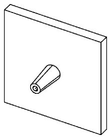
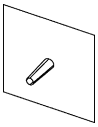
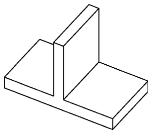
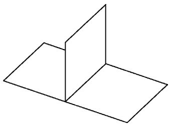
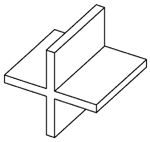
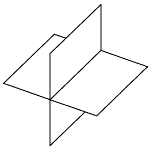
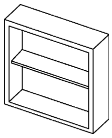
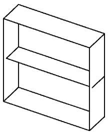
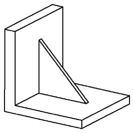
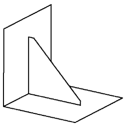


Fig. 3. Object model representation of mid-surface model.

Table 1
Summary of feature types and graph characteristics (Reproduced by permission of ASME) [15]

Feature class	Feature type	Solid model	Mid-surface model	Manufacturability impact
Face feature	Fin			Fins may affect the external wall quality at the attachment. Careful design of fin proportions can minimise problems [13]
	Hole			Small holes may be more economic to drill, may cause weld lines [13]
	Boss			The proportions of bosses are important for main wall quality. Supporting ribs may be required to react lateral forces [14]
Junction features	T-junction			High order junctions may cause sink marks or warping. Wall thickness proportions and angles are important [13]
	X-junction			High order junctions may cause sink marks and warping. Where possible should be redesigned as two staggered junctions [13]
Stiffener features	Rib			Ribs may cause warping or appearance problems—rib proportions are important [13]
	Buttress			Buttresses may cause warping or appearance problems—proportions are important [13]

therefore to identify the sub-types of wall features in a part, and to find the intersections and junctions between those features. The features that need to be recognised from the mid-surface model have been categorised into three main types—face features, junction features, and stiffener features. These feature types are shown in Table 1, along with their effect on manufacturability.

The features can be recognised from their AMAG characteristics. Table 2 shows the mid-surface geometry and associated graph segment for the feature classes from Table 1. The graph segments are shown with two attributes associated with each graph arc. The first attribute identifies whether the edge is connected to an internal or external edge loop within the face (0 for external and 1 for internal), and the second attribute identifies how many times the edge is used in the face (for example if an edge extends into the interior of a face it will be used twice in the edge loop for that face). For clarity the edge-loop identifiers are not shown in this figure.

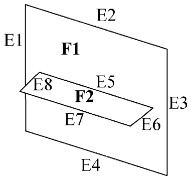
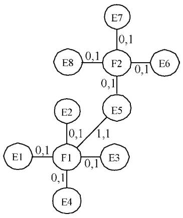
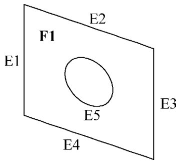
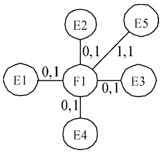
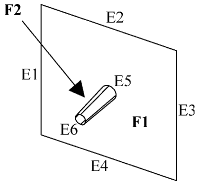
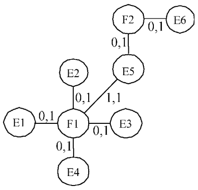
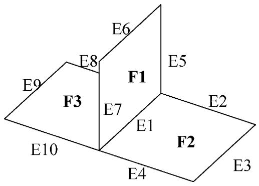
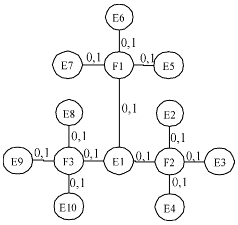
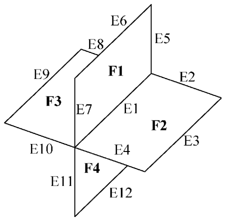
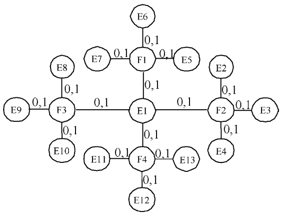
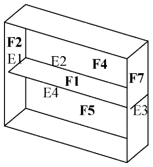
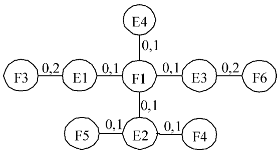
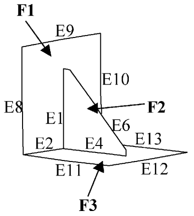
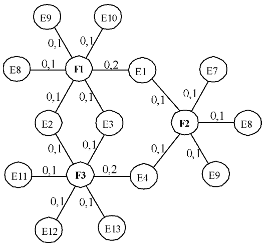
4.2. Feature recognition

The feature recognition process is undertaken in three stages—firstly the mid-surface model is parsed to construct a face-edge adjacency matrix that represents the AMAG, then the feature recognition algorithms perform an initial feature identification based on topology, and finally the feature recognition is completed by performing geometry checks using the mid-surface model.

4.2.1. Face-edge adjacency matrix

Adjacency matrices can be used to represent graph structures for feature recognition from a CAD solid model [1]. The FAG for a manifold solid model with n faces can be represented using an $n \times n$ face adjacency matrix, where a 1 in the matrix indicates the existence of an edge connecting the two faces, and a 0 indicates that two faces are not connected. The face adjacency matrix assumes that each edge in the model connects exactly two faces (a requirement for a manifold solid).

Table 2
AMAGs for simple feature types

Feature class	Feature type	Mid-surface geometry	AMAG	Graph characteristics
Face feature	Fin			A *fin feature* is recognised by the existence of a single internal edge that is connected to one edge of another face. The connected face must be otherwise unconnected (e.g. F2)
	Hole			A *hole feature* is recognised by the existence of an internal edge loop that is not connected to any other faces (e.g E5)
	Boss			A *boss feature* is recognised by an internal edge that is connected to a cylindrical or conical face by a circular edge
Junction features	T-junction			A *T-junction feature* is recognised by an edge that is used by three faces (e.g. E1). An attribute representing the angles between the faces is also required to complete the feature recognition
	X-junction			An *X-junction feature* is recognised as an edge that connected to four adjacent faces. The angles between the face normals should be 90° to fully identify the X-junction
Stiffener features	Rib			A *rib feature* is recognised as a face with three or more adjacent edges connected at order 3 or higher and at least one unconnected edge. (e.g. F1)
	Buttress			A *buttress feature* is recognised as a face with two adjacent edges connected at order 3 or higher, and the remaining edges unconnected (e.g. E1 and E4)

In the AMAG both faces and edges are represented as nodes in the graph, so a *face-edge* adjacency matrix is required to represent the geometry graph. The complete *face-edge* adjacency matrix for a model with m edges and n faces is an $(n+m)\times(n+m)$ *matrix*, but since only the face to edge connectivity is required only an $n\times m$ a subset of the matrix is required to capture the graph topology.

The *face-edge* adjacency matrix for the T-shaped part in Fig. 2(a) is the 3×10 matrix shown in Fig. 4, which is a subset of the complete *face-edge* adjacency matrix for

$$\begin{array}{c|cccccccccc|} & E1 & E2 & E3 & E4 & E5 & E6 & E7 & E8 & E9 & E10 \\ \hline F1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ F2 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ F3 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{array}$$

Fig. 4. Face-edge adjacency matrix, for T-shaped part.

the T-shaped part, that would be a 13×13 matrix. The values in the matrix m represent the connectivity between the faces and edges in the graph, where $m_{ij}=1$ means that edge j is a bounding edge of face i, and $m_{ij}=0$ means that edge j is not a bounding edge of face i.

4.2.2. Face-edge adjacency matrix attributes

The *face-edge* adjacency matrix that is constructed in the feature recogniser stores three attributes for each face-edge adjacency forming a three-dimensional matrix. The attributes that are stored are:

- the edge loop id number
- an identifier for the edge loop within the current face (for outer edge loop = 0)
- the number of times the edge is used by the face.

Once the adjacency matrix has been constructed it can be used to evaluate some further properties of the model. The order of each edge (i.e. the number of faces connected to the edge) can be obtained by counting the number of non-zero values in an edge column. The number of edge loops in a face can be obtained by counting the number of edge loops in a face row. These properties of the adjacency matrix are used to aid the feature recognition process.

4.2.3. Algorithms for feature recognition

4.2.3.1. Face features. Face features are classified as features that are attached to the interior of a face in the part and may be holes, slots, fins, bosses or other attached features. The graph segment for a simple hole feature can be seen in Section 4.1, Table 2. The face feature recognition algorithm is able to identify features that are completely enclosed within a single face of the part, but future work will extend the approach to identify features that are connected to both the boundary and interior of a face. Once the existence of a face feature has been identified, further geometric processing is required to recognise the type of face feature. For example a boss is recognised by a circular edge connecting a face feature to an internal edge loop on its parent face. The algorithm for face feature recognition is shown below, and takes as its input a list of faces with internal edge loops, and a list of the orders of each edge, both of which can be obtained from the adjacency matrix.

```
Algorithm Find_Face_Features
    begin
        Let FACES-WITH-INTERNAL-LOOPS be a list of all the faces in the
        model with internal edge loops
        for each face f in FACES-WITH-INTERNAL-LOOPS do
            for each internal loop l in f do
                ATT-FACES[l] = {} ;list of attached faces for each loop
                for each edge e in l do
                    if EDGE-ORDERS[e] > 1 then
                        Add adjacent faces to ATT-FACES[l]
                end
            end
        if ATT-FACES[l] = {} then
            Edge-loop l connects to a HOLE
        else if length (ATT-FACES[l] = 1) & (edge type = POLYLINE) then
            edge-loop l connects to a FIN
        else if (edge type = CIRC_ARC) & (face type = CYL or CONICAL) then
            edge-loop l connects to a BOSS
        else
            edge-loop l connects to an UNKNOWN-FACE-FEATURE with
            attached faces ATT-FACES[l]
        end
    end
```

4.2.3.2. Stiffener features. Stiffener features such as ribs and buttresses are structural features that are used to add strength to the part. The stiffener features are identified from the AMAG as faces with specific patterns of adjacent high order and unconnected edges. The graph segment for a simple stiffener feature can be seen in Section 4.1, Table 2. The order of each edge can be obtained from the attributed adjacency matrix and the ordered list of edges in each edge loop is read from the mid-surface model.

```
Algorithm Find_Stiffeners
  begin
    for each face f do
      let oel be the identifier for the outer edge loop
      let LIST-OF-EDGES[oel] be an ordered list of the edges in oel
      NO-OF-ADJ-HO-EDGES = 0 ; adjacent high-order edges
      MAX-ADJ-HO-EDGES = 0; max no. of adjacent high-order edges in loop
      NO-OF-UNCON-EDGES = 0 ; unconnected edges
      for each edge e in LIST-OF-EDGES[oel] do
        if EDGE-ORDERS[e] >= 3 then
          NO-OF-ADJ-HO-EDGES = NO-OF-ADJ-HO-EDGES + 1
          if NO-OF-ADJ-HO-EDGES > MAX-ADJ-HO-EDGES then
              MAX-ADJ-HO-EDGES = NO-OF-ADJ-HO-EDGES
          end
        else if EDGE-ORDERS[e] = 2 then
          NO-OF-ADJ-HO-EDGES = 0
        else if EDGE-ORDERS[e] =1 then
          NO-OF-UNCON-EDGES = NO-OF-UNCON-EDGES+ 1
          NO-OF-ADJ-HO-EDGES = 0
        end
      end
      if (MAX-ADJ-HO-EDGES = 2) & (NO-OF-UNCON-EDGES >=1) then
          face f is a buttress feature
      else if (MAX-ADJ-HO-EDGES >= 3) & (NO-OF-UNCON-EDGES >=1) then
          face f is a rib feature
      end
  end
```

4.2.4. Geometric checking

The topological feature identification described above is supported by further geometric checks to validate and categorise the features that have been recognised. The geometric checks that are performed for the various feature classes are summarised below.

4.2.4.1. Face features. Face features are initially identified from their face-edge connectivity, then the geometric characteristics of the feature's edges and faces are used to complete the feature recognition. For example a boss is recognised as cylindrical or conical face feature that is connected to its parent face by a circular arc. A linear fin feature is recognised as a planar face feature that is connected to its parent face by a linear edge.

4.2.4.2. Junction features. Junction features are the building blocks for stiffener features. The order of a junction can be recognised from the part topology, but the angles between the faces at the junction are also important for moulding features. The angle between faces is calculated by computing the surface normal for each face, and then calculating the angle between the surface normals. In the current implementation the angle checks are only performed for planar faces.

4.2.4.3. Stiffener features. Stiffener features are primarily identified topologically, but the angles between faces of high order junctions are used to validate the stiffener features.

5. Implementation

A demonstrator for the feature recogniser has been developed and tested for a range of simple parts. Fig. 5 shows a flow chart of the feature recognition process. The input to the process is a B-Rep solid model of the part generated using a CAD solid modeller. Step 1 is the pre-processing step in which a mid-surface representation of the part is generated from solid model and exported as a STEP AP203 file. In Step 2 the mid-surface geometry is read from the STEP file and used to populate the mid-surface model. In Step 3 the attributed adjacency matrix is constructed from the mid-surface model, and in Step 4 the feature recognition is performed. The results are presented to the user graphically as a coloured STEP part 21 file that can be read into their CAD system, and a text report on the screen.

The demonstrator has been implemented in GNU C++ on the Cygwin platform, and using the data modelling standard STEP for data exchange. The pre-processing step that generates the mid-surface model has been implemented using EDS I-DEAS NX Series [12], but any alternative mid-surfacing tool could be substituted as long as it is able to

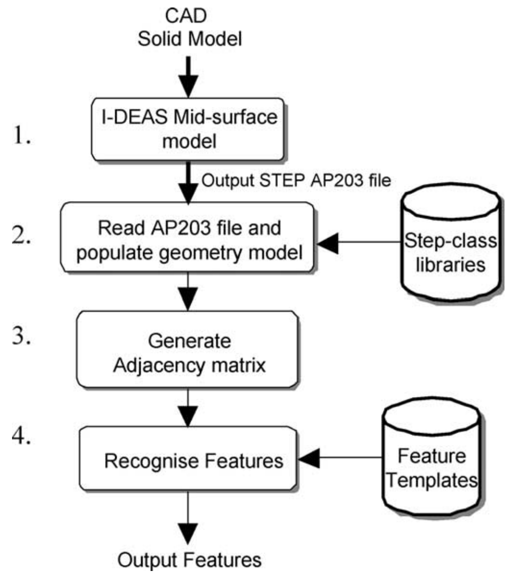


Fig. 5. Flow chart of feature recognition process.

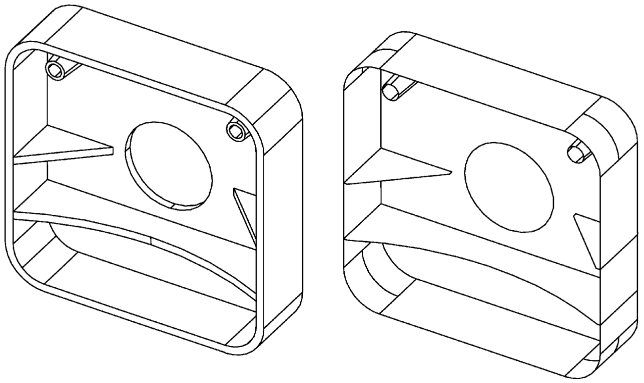


Fig. 6. Solid model and mid-surface for a simple plastic part.

output the mid-surface geometry in the required format (STEP AP203 Part 21 file, Manifold Surfaces with Topology representation). The mid-surface model and feature recognition algorithms have been developed as a standalone application using object oriented C++, with the STEP Class Libraries from National Institute of Standards and Technology (NIST) [16] used to develop the STEP interface.

The STEP AP203 file is parsed to extract topology and geometry information and populate the mid-surface model. The STEP geometry representation has an eight-level hierarchy comprising open_shell, shell_based_surface_model, face_surface, face_bound, edge_loop, oriented_edge, edge_curve and vertex_point entities. In the mid-surface model the representation is simplified to a four-level hierarchy (midsurface_model, face, edge_loop, edge) with additional information captured as attributes of the geometry entities. The attributed adjacency matrix that represents the MAG is then constructed from the mid-surface model and stored as a three-dimensional array of faces, edges, and attributes. The feature recognition algorithms make use of the adjacency matrix and mid-surface model when recognising features.

The STEP part 21 files require a small amount of preprocessing to overcome limitations in the NIST STEP toolkit. This pre-processing is performed automatically using an awk script, and the whole feature recognition process is initiated through a UNIX shell script.

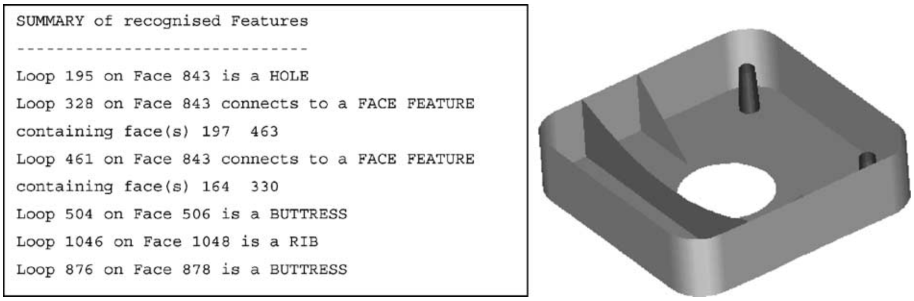
5.1. Case study 1—simple plastic part

The feature recognition process has been tested for a range of moulded parts. The first case study is an idealised part with some common injection moulding features, which has been used to demonstrate the basic principles of the feature recogniser. Fig. 6 shows the CAD solid model and mid-surface abstraction for the part. It can be clearly seen that the mid-surface model maintains the important geometric characteristics of the part, including the main part wall, and attached features.

The feature recogniser constructs a mid-surface model with 21 faces, 24 edge loops and 57 edges, and a 21×57×3 array representing the adjacency matrix. The results of the feature recognition process are generated as a text report and a colour coded STEP file. The results for the simple plastic part are shown in Fig. 7 below.

5.2. Case study 2—remote control casing

The second case study is a more realistic part representative of the type used in the domestic consumer products. The model is of one half of the casing of a remote control unit and contains several common moulding features (Fig. 8).



```
SUMMARY of recognised Features
------------------------------
Loop 195 on Face 843 is a HOLE
Loop 328 on Face 843 connects to a FACE FEATURE
containing face(s) 197  463
Loop 461 on Face 843 connects to a FACE FEATURE
containing face(s) 164  330
Loop 504 on Face 506 is a BUTTRESS
Loop 1046 on Face 1048 is a RIB
Loop 876 on Face 878 is a BUTTRESS
```

Fig. 7. Feature recognition results for simple plastic part.

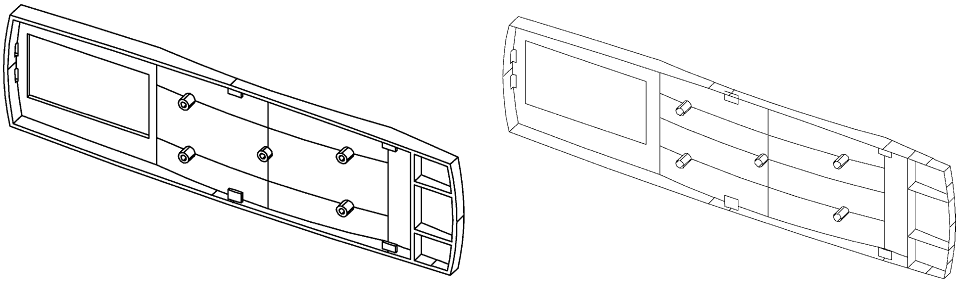


Fig. 8. Solid model and mid-surface model for remote control part.

The mid-surface model contains 33 faces, 43 edge loops and 99 edges and is more complicated than the first example because it has intersecting rib features, a curved surface and a face with multiple features. The mid-surface generation was performed automatically within the I-DEAS software, although a small number of unwanted surfaces had to be deleted manually from the mid-surface prior to feature recognition.

The feature recogniser identified one hole, five intersecting ribs, four fins, and five bosses, and the graphical output of the feature recognition process is a feature report and colour coded STEP model shown in Fig. 9.

The results for this part are generally good, although they also show some limitations in the current version of the demonstrator. The two small holes at the end of the part are not recognised because their boundaries are formed from the combination of edge loops on two adjacent faces, and the hole recognition algorithm expects a hole to be attached to a single face.

6. Summary and conclusions

In this paper a novel feature recognition approach for moulded parts has been presented based on a mid-surface approach. The main contribution of the work has been the application of feature recognition techniques to mid-surface models to allow the recognition of moulding features on thin walled parts. A new AMAG has been developed to represent mid-surface geometry, and a feature recognition methodology for moulding features has been implemented. The techniques that have been developed in the research build on existing graph-based feature recognition techniques that have previously been applied to machining feature recognition for prismatic solid models. [1,6,7] However, the existing techniques have been extended to support feature recognition for a new application area of moulded parts, and have required the development of a methodology that is applicable to non-manifold geometries.

The methodology presented in this paper has several advantages over other methods for moulded parts. Firstly, using the mid-surface as a basis for feature recognition makes the topology relationships required for thin walled feature recognition much more accessible than they are on a CAD solid model, and the relationship back to the CAD solid model geometry can be maintained. Secondly, using the mid-surface significantly reduces the number of feature interactions that need to be computed to identify thin wall features that are of interest for moulding evaluation, and thirdly the methodology is applicable to parts with complex curved surfaces in the main wall (although the current implementation of the demonstrator does not fully validate some feature types such as ribs if they have curved faces because the geometric checking has only been implemented for planar faces).

Most existing feature recognition techniques are aimed at machining features and assume that the part can be recognised as a base stock with machining features removed from it, but this approach is not suited to feature recognition for a moulded part that is entirely composed of intersecting thin walled sections. The mid-surface approach that is presented here provides a more promising basis for moulding feature recognition than existing techniques.

A demonstrator for the feature recogniser has been implemented using C++ and ISO 10303 STEP. The demonstrator has been tested on a variety of geometry models and produced good results for a range of moulded parts. The algorithms that have been developed are robust for a range of geometries and have been tested on parts with up to 60 mid-surface faces. Several limitations with the existing demonstrator have been identified and these will be investigated in future work:

(1) The methodology that has been developed is dependent on the use of third party mid-surfacing tools to generate the mid-surface representation of the part prior to feature recognition, but the mid-surface is well established as an abstraction for finite element analysis and flow

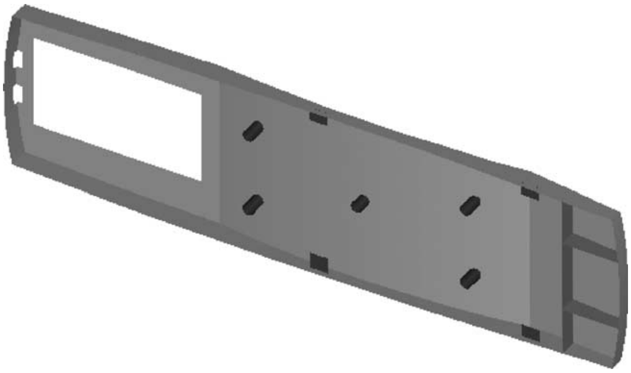


Fig. 9. Results of feature recognition process for remote control part.

simulation on thin walled objects. It is acknowledged that there is not yet a fully robust mid-surfacing algorithm, but an existing commercial implementation has been found to give reasonable results for a variety of parts. The current generation of mid-surfacing algorithms work well for thin walled parts that have a relatively small variation in wall thickness, and these characteristics are well aligned with the requirements of moulding manufacturing process, and mid-surfacing algorithms are currently being implemented in a number of CAE tools.

(2) The current implementation of the feature recogniser recognises only a limited number of moulding features, and the geometric checking is relatively simplistic. Future work will extend the approach to identify additional feature types, and to refine the geometric validation of the features that are recognised.

One current limitation is that the algorithms assume that each face on the part is represented by a single surface in the CAD model, but for some parts this assumption is not true and the feature recognition algorithms may not recognise the moulding features correctly. This limitation could be overcome by the implementation of a 'face group' function that would group together tangentially connected faces prior to feature recognition.

Future work could also develop algorithms to recognise features that are more topologically complex than those currently supported. For example parts in which the faces are connected together to form loops of faces or features that form bridges from the interior of one face to the interior of another could be recognised using extensions to the feature recognition algorithms.

(3) At present the output from the feature recogniser is in the form of a colour-coded STEP AP203 file of the mid-surface geometry, and a text report. A future objective is to map the recognised features back onto the original CAD solid geometry so that the features can be associated with the original CAD geometry as well as the mid-surface abstraction.

(4) A knowledge-based manufacturing advisor for moulded parts is currently under development that will be integrated with the feature recogniser to assist inexperienced designers in the design of products for injection moulding or other near-net shape manufacturing processes.

References

[1] Shah J, Mantyla M. In: Parametric and feature based CAD/CAM. London: Wiley; 1995 p. 364.

[2] Knight B, Cowell D, Preddy K. An object-oriented support tool for the design of casting procedures. Eng Appl Artif Intell 1995;8(5):561–7.

[3] Rosen DW, Dixon JR, Finger S. Conversions of feature-based design representations using graph grammar parsing. J Mech Des 1997;116: 785–92.

[4] Lu SC, Rebello AB, Miller RA, Kinzel GLA. Volume-based geometric reasoning approach for diecastability evaluation. Proceedings of the Computers in Engineering Conference and the Engineering Database Symposium. American Society of Mechanical Engineers; 1995.

[5] Ravi B, Srinivasan MN. Hot spot detection and modulus computation by computer graphics. 56th World Foundry Congress p. 12.1–12.8.

[6] Joshi S, Chang TC. Graph-based heuristics for recognition of machined features from a 3D solid model. Comput Aided Des 1988;20(2):58–66.

[7] Gao S, Shah J. Automatic recognition of interacting machining features based on minimal condition subgraph. Comput Aided Des 1998;30(9):727–39.

[8] Donaghy RJ, Armstrong CG, Price MA. Dimensional reduction of surface models for analysis. Eng Comput 2000;16:24–35.

[9] Sheehy DJ, Armstrong CG, Robinson DJ. Shape description by medial surface construction. IEEE Trans Visual Comput Graphics 1996;1(1): 62–72.

[10] Blum H. A transformation for extracting new descriptors of shape. In: Wathen-Dunn W, editor. Models for the perception of speech and visual form. Cambridge, MA: MIT Press; 1967, p. 362–81.

[11] Price M, Stops C, Butlin G. Medial object a medial object toolkit for meshing and other applications. International Meshing Roundtable Conference at Sandia National Laboratories, Albuquerque NM, USA1995.

[12] EDS I-DEAS NX Series User Guide. EDS; 2002.

[13] Boothroyd G, Dewhurst P, Knight W. Product design for manufacture. New York: Marcel Dekker Inc; 2002.

[14] Ticona. Designing with plastics, The fundamentals. Ticona Ltd; 2000.

[15] Lockett HL, Guenov MD. An intelligent manufacturing advisor for casting and injection-moulding based on a mid-surface approach. Proceedings of DETC'02 ASME 2002 Design Engineering Technical Conferences and Computers and Information in Engineering Conference. Montreal, Canada, September 29–October 2, 2002.

[16] National Institute of Standards and Technology—STEP Class Libraries. Available from: http://www.nist.gov/scl [accessed March 2002].

Mrs Helen Lockett is a lecturer in Computer Aided Design at the School of Engineering, Cranfield University. She has a BSc in Electronic Engineering from the University of Sussex and is currently registered as staff PhD candidate at Cranfield University. Helen teaches computer aided design and integrated product development to post-graduate Aeronautical engineering students. Her research interests are in design for manufacture, intelligent design tools and computational methods for integrated product development. She has contributed to major European Union funded research projects in aircraft multidisciplinary design, and has also undertaken consultancy projects in computer aided design and design for manufacture.

Dr. Marin D. Guenov is a Senior Lecturer (Advanced Engineering Methods) in the School of Engineering, Department of Power Propulsion and Aerospace Engineering at Cranfield University, UK. He holds MEng in Mechanical Engineering and a PhD in Materials Handling Systems and Operational Research. Currently Dr Guenov leads national and international multidisciplinary research projects supported by the European aerospace industry in the areas of design of complex systems, modelling and simulation for synthetic environments, multi-disciplinary design analysis and optimization, and infrastructures for collaborative design. Dr. Guenov is a member of the Royal Aeronautical Society, The Association of Cost Engineers and is a Charted Engineer.

APPENDIX B - Manufacturing Knowledge Tables

Manufacturing Facts

ID	Process	Material	Feature	Fact	Source
SCF1	Sand Casting			Minimum wall thickness = 6.35 mm	1
SCF2	Sand Casting			Maximum Wall thickness = 127 mm	1
SCF3	Sand Casting			Main wall draft angle = 2°	1
SCF4	Sand Casting		Rib Buttress Boss	Attached Features draft angle = 1°	1
SCF5	Sand Casting			Section Change Taper Ratio = 4:1	1
SCF6	Sand Casting	Aluminium	Rib	Minimum wall thickness = 2.54mm	1
SCF7	Sand Casting	Magnesium Alloys	Rib	Minimum wall thickness = 3.3mm	1
SCF8	Sand Casting	Steel	Rib	Minimum wall thickness = 3.3mm	1
SCF9	Sand Casting	Aluminium		Minimum wall thickness = 4.75 mm	2
SCF10	Sand Casting	Copper Alloys		Minimum wall thickness = 2.3 mm	2
SCF11	Sand Casting	Gray Cast Iron		Minimum wall thickness = 3.0 mm	2
SCF12	Sand Casting	Steel		Minimum wall thickness = 5.0 mm	2
SCF13	Sand Casting	Magnesium Alloys		Minimum wall thickness = 4.0 mm	2
SCF14	Sand Casting	Malleable Irons		Minimum wall thickness = 3.0 mm	2
SCF15	Sand Casting	Any	Rib	Rib Thickness/ External Wall Thickness Ratio = 0.8	6
DCF1	Die Casting			Draft Angle = 0.25 - 0.75°	3
DCF2	Die Casting	Aluminium Alloys		Maximum Wall thickness = 0.9 mm	3
DCF3	Die Casting	Aluminium Alloys		Minimum Wall thickness = 5.08 mm	6
DCF4	Die Casting	Aluminium Alloys		Minimum Draft Angle = 0.5°	3
DCF5	Die Casting	Zinc Alloys		Minimum Wall thickness = 0.6 mm	3
DCF6	Die Casting	Zinc Alloys		Minimum Draft Angle = 0.25°	3
DCF7	Die Casting	Copper Alloys		Minimum Wall thickness = 1.25 mm	3
DCF8	Die Casting	Copper Alloys		Minimum Draft Angle = 0.7°	3
IMF1	Injection Moulding		Rib	Rib thickness <= 50% of main wall thickness	4
IMF2	Injection Moulding		Rib	Max rib height = 3 x main wall thickness	4
IMF3	Injection Moulding		Buttress Boss	Minimum rib spacing = 2 x main wall thickness	4

IMF4	Injection Moulding		Buttress	Minimum attached edge length = 2 x main wall thickness	4
IMF5	Injection Moulding		Rib Buttress Boss	Radius a rib ends = 25 - 40% of Main Wall Thickness	4
IMF6	Injection Moulding		Rib	Minimum Draft Angle = 0.5°	4
IMF7	Injection Moulding		Buttress Boss	Thickness = 50 - 70% of main wall thickness	4
IMF8	Injection Moulding			Minimum wall thickness = 0.8 mm	5
IMF9	Injection Moulding			Maximum Wall thickness = 4.8 mm	5
IMF10	Injection Moulding			Minimum Main wall draft angle = 0.5°	5
IMF11	Injection Moulding			Recommended Draft angle = 1.5 - 3 degrees	5
IMF12	Injection Moulding			Section Change Ratio = 3:1	5
IMF13	Injection Moulding	GE Cycoloy		Minimum wall thickness = 1.2mm	6
IMF14	Injection Moulding	Acetal		Typical Wall thickness range = 0.8 mm - 3.6 mm	5
IMF15	Injection Moulding	Acrylic		Typical Wall thickness range = 0.6 mm - 3.0 mm	5
IMF16	Injection Moulding	Long-fibre reinforced plastics		Typical Wall thickness range = 1.9 mm - 25.4 mm	5

Sources:

1 http://www.engineersedge.com/sand_cast.htm
2 http://www.efunda.com/processes/metal_processing/sand_casting_table.cfm
3 http://www.efunda.com/processes/metal_processing/die_casting.cfm
4 http://www.geplastics.com/resins/global/pdf/europe_guides/desineng.pdf
5 Designing with Plastic, The fundamentals. Ticona
6 http://www.geplastics.com/resins/techsolutions/PDFs/cycoloy_2004.pdf
7 Handbook of product design for manufacturing : a practical guide to low-cost production, J. G. Bralla

Manufacturing Rules							
ID	Process	Material	Feature	Knowledge	Rule	Explanation/ Advice	Source
GENR1	Any-Moulding	Any	Any	Parts should be designed with as uniform wall thickness where possible	IF: 1)Process = Any-Moulding, and 2) Wall-Thckness-Variation > 10% THEN: Non-uniform-wall-thickness = True, if possible redesign with more uniform wall thickness	wall thickness variations may cause warping and appearance defects	1,2,3
GENR2	Any-Moulding	Any	Any	If the part is too thick and strength is not a consideration then the wall thickness should be reduced	IF: 1) Process = Any-Moulding, and 2) Wall-Thickness > Max-Wall-Tk, and 3) Strength is not important THEN: Reduce wall thickness to less than max-wall-Thickness	Thick walls increase cycle time and can cause appearance defets	4
GENR3	Any-Moulding	Any	Any	If the part is too thick and strength is a consideration then the part should be redesigned with a thinner section and ribs to increase strength	IF: 1)Process = Any-Moulding, and 2) Wall-Thickness > Max-Wall-Tk, and 3) Strength is important THEN: Consider redesign with thinner wall thickness and using ribs to increase strength	The strength of moulded parts can be increase by use of ribs, while maintaining a smaller wall thickness	4
GENR4	Any-Moulding	Any	Any	If the part is too thick and strength is a consideration then the part should be redesigned with a thinner section and using a material with improved mechanical properties	IF: 1) Process = Any-Moulding, and 2) Wall-Thickness > Max-Wall-Tk, and 3) Strength is important, and 4) Material is unknown THEN: Consider redesign with thinner wall thickness and using ribs to increase strength	The strength of moulded parts can be increased by selecting a material with improved materials properties	4

GENR5	Any-Moulding	Any	Any	Parts must be designed with adequate draft angle to facilitate removal from the mould	IF: 1) Process = Any-Moulding
					THEN: Minimum Draft Angle = Min-Draft-Angle
							1,2,3

GENR6	Any-Moulding	Any	Any	The design should not have abrupt section changes, where a section change is required a gradual taper must be applied	IF: 1) Process = Any Moulding, and 2) Non-uniform-wall-thickness = True THEN: Set the taper angle at section change to at least min-section-change-ratio	Wall thickness variations influence cooling rates and may result in warping and appearance defects	1,2,3
GENR7	Any-Moulding	Any	Any	All corners should be radiussed	IF: 1) Process = (Any Moulding) THEN: Set the minimum corner radius to be at least min-corner-raduis	Corner radii are required to avoid stress concentrations	1,2,3
SCR1	Sand Casting	Any	Rib	Rib thickness should be proportionately thinner than main wall thickness	IF: 1) Process = (Sand Casting), and 2) Feature = (Rib) THEN: Rib wall thickness should be proportinately smaller than external wall thickness	Internal sections cool more slowly than external walls	6
SCR2	Sand Casting	Any	High Order Junction	The number of intersecting ribs at one point should be minimised to avoid hot-spots	IF: 1) Process = (Sand Casting), and 2) Feature = (HO-Junction) 3) Junction order = 4 THEN: If possible replace X-junction with two staggered T-junctions	High Order junctions can cause hot-spots in casting	6
SCR2	Sand Casting	Any	High Order Junction	The number of intersecting ribs at one point should be minimised to avoid hot-spots	IF: 1) Process = (Sand Casting), and 2) Feature = (HO-Junction) 3) Junction order > 4 THEN: If possible put a cored hole at the intersection to speed cooling	High Order junctions can cause hot-spots in casting	6

DCR1	Die Casting		Boss	Projections and bosses can be difficult to fill: buttresses assist flow to such features and strengthen the component	IF: 1) Process = Die-Casting, and 2) Feature = Boss THEN: Consider using buttresses to facilitate material flow		5
DCR2	Die Casting		Rib	Ribs should not be square in section.	IF: 1) Process = Die-Casting, and 2) Feature = Rib THEN: Design Ribs with blended sections to facilitate die-filling	Blended sections and curved buttresses aid die filling	5
DCR3	Die Casting		Rib	Avoid thick sections at intersections of ribs	IF: 1) Process = Die-Casting, and 2) Feature = Rib, and 3) Rib-Tk > max-rib-prop * wall-thickness THEN: Reduce Rib thickness to less than max-rib-prop * wall-thickness		5
DCR4	Die Casting		Hole	Blind holes are preferable to through holes,	IF: 1) Process = Die-Casting, and 2) Feature = Hole, and 3) Type = Through THEN: Consider redesigning hole as blind hole	Through holes can cause problems with flash	5
DCR5	Die Casting		Hole	Holes should be tapered	IF: 1) Process = Die-Casting, and 2) Feature = Hole THEN: Ensure that hole draft angle is at least min-draft-angle	Tapered holes assist with removal of the casting from the die	5
IMR1	Injection Moulding		Rib Buttress	Projections from the main wall should be designed with a proportionally smaller wall thickness than the main wall	IF: 1) Process = Injection-Moulding, and 2) Feature = Rib, and 3) Rib-Tk > max-rib-prop * wall-thickness THEN: Reduce Rib thickness to less than max-rib-prop * wall-thickness	This will facilitate cooling	3
IMR2	Injection Moulding		Boss	Projections from the main wall should be designed with a smaller thickness than the main wall	IF: 1) Process = Injection-Moulding, and 2) Feature = Boss, and 3) Boss-Tk > max-boss-prop * wall-thickness THEN: Reduce Boss thickness to less than max-boss-prop * wall-thickness	This will facilitate cooling	3
IMR3	Injection Moulding		Boss Rib Buttress	If appearance is important then the thickness of protrusions from the main wall should be minimised	IF: 1) Process = Injection-Moulding, and 2) Feature = Boss, and 3) Boss-Tk > max-boss-prop * wall-thickness, and 4) Appearance is important THEN: Feature thickness should be minimised		3
IMR4	Injection Moulding		Buttress	Buttresses must be designed with appropriate spacing			3
IMR5	Injection Moulding		Rib Buttress	The length of the buttress attachment face must be designed with regard to the main wall thickness			3
IMR6	Injection Moulding		Rib buttress	Ribs must have generous radii			3
IMR7	Injection Moulding		Rib Buttress	Ribs must be designed with appropriate draft angle	IF: 1) Process = Injection-Moulding,and 2) Feature = Rib THEN: Ensure that rib draft angle is at least min-draft-angle		3

IMR8	Injection Moulding		Boss	To increase strength of a boss without increasing thickness use buttress supports or attach boss to a nearby wall.	IF: 1) Process = Injection-Moulding, and 2) Feature = Boss, and 3) Strength is important THEN: Consider using buttresses to give extra strength to boss		3
IMR9	Injection Moulding			Where changes in thickness are involved take care that the direction of melt flow is from thick to thin	IF: 1) Process = Injection-Moulding, and 2) Non-uniform-wall-thickness = True THEN: Ensure that the directionof melt flow is from thick to thin		3

Source:

Helen Lockett
Updated 20th December 2004

1 http://www.engineersedge.com/sand_cast.htm
2 http://www.efunda.com/processes/metal_processing/die_casting.cfm
3 http://www.geplastics.com/resins/global/pdf/europe_guides/desineng.pdf
4 Designing with Plastic, The fundamentals. Ticona
5 The DieCasting Book. Street, A. C.
6 Handbook of product design for manufacturing : a practical guide to low-cost production, J. G. Bralla. 1986, McGraw Hill
7 Foundry Technology, P. Beeley. 2001, Butterworth-Heinemann.

APPENDIX C - Software Source Code

Midsurf.h

Midsurf.cc

Geomfromstep.cc

Manufacturing-advisor.clp

```
/*
 * midsurf.h
 *
 * Helen Lockett, Cranfield University
 *
 * June  2005
 *
 * h.lockett@cranfield.ac.uk
 *
 * Developed using NIST STEP Class Library
 *
 * Available for download from
 *
 * http://www.mel.nist.gov/msidstaff/sauder/SCL.htm#download
 *
 * Some sections adapted from SCL basic examples
 *
 * In particular treg.cc, Written by Ian Soboroff, NIST. June, 1994
 *
 */

#include <vector>
#include "math.h"
#include <string>

#ifndef VERTEX_SPEC
#define VERTEX_SPEC


class Vertex {
public:
   Vertex();
   void add_vertex(const int, const float, const float, const float);
   int get_vertex_id();
   float get_coord_component(const int);
private:
   int vertex_id;
   vector <float> coords;
};

#endif

#ifndef EDGE_SPEC
#define EDGE_SPEC

class Edge {
public:
   Edge();
   void add_edge(const int);
   void print_edge_label();
   void add_loop_to_edge(const int);
   int get_edge_id();
   void print_list_of_loops();
   int get_no_of_loops();
   void add_face_to_edge(int);
   int get_no_of_faces_using_edge();
   int get_face_id(const int);
private:
   int edge_label;
   int list_of_loops[10];
   int no_of_loops;
   char parent_part[15];
   int edge_number ;
   int no_of_faces_using_edge;
   int faces_using_edge[10];
};

#endif

#ifndef LOOP_SPEC
#define LOOP_SPEC

class Edge_Loop {
public:
   Edge_Loop();
```

```
   void add_edge_loop(const int );
   void print_list_of_edges();
   int get_no_of_edges();
   int get_edge(int);
   void add_edge_to_loop(const int);
   int get_edge_loop_id();
   void add_attached_face(const int);
   int get_attached_face(const int);
   int get_no_of_attached_faces();
private:
   int edge_loop_label;
   int no_of_edges;
   int list_of_edges[20];
   int list_of_adjacent_faces[10];
   int attached_faces[100];
   int no_of_attached_faces;
};

#endif

#ifndef BOUND_SPEC
#define BOUND_SPEC

class Bound {
public:
   Bound();
   void add_bound(const int );
   void add_loop_to_bound( const int);
   int get_bound_id();
   int get_edge_loop_name();
private:
   int bound_label;
   int edge_loop_name;
};

#endif
/*
#ifndef OUTER_BOUND_SPEC
#define OUTER_BOUND_SPEC

class Outer_Bound {
public:
   Outer_Bound();
   void add_outer_bound(const int );
   void add_edge_outer_bound(const int);
   int get_outer_bound_id();
   int get_no_of_edges();
   int get_edge(const int);
private:
   int outer_bound_label;
   int no_of_edges;
   int list_of_edges[20];
   int list_of_adjacent_faces[10];
};

#endif

*/
#ifndef FACE_SPEC
#define FACE_SPEC

class Face {
public:
   Face();
   void add_face(int );
   void add_bound(const int);
   void add_edge(const int);
   void add_edge_to_bound(const int, const int);
   int get_face_id();
   int get_bound_id(const int);
   int get_bound_no_from_id(const int);
   void find_faces_adjacent_to_edge();
   int get_no_of_bounds();
   int get_bound(int);
   void add_face_geometry(const int);
   int get_surface_geom_id();
   void add_vertex_to_face(const int);
   void set_planar();
   void set_cylindrical();
   int get_vertex_id(const int);
   void set_normal_dir(const float, const float, const float);
   float get_normal_dir_component(const int);
   int get_planar();
   int get_cylindrical();
   std::string get_ideas_id();
   void set_ideas_id(const std::string);
private:
   Bound bound[20];
//   Outer_Bound outer_bound;
   int no_of_bounds;
   int no_of_edges;
   int list_of_edges[20];
   int list_of_adjacent_faces[10];
   int list_of_bounds[20];
   int face_bound_no;
   int face_label ;
   int bound_label;
   int planar;
   int cylindrical;
   int face_geometry;
   std::string ideas_id;
   vector<int> vertices;
   int no_of_vertices_in_face;
   vector<float> normal_dir;
};

#endif

#ifndef MIDSURFACE_SPEC
#define MIDSURFACE_SPEC

class MidSurface {
public:
   MidSurface();
   void print_face_label(const int );
   int midsurface_label() const;
   void add_face(const int);
   void add_bound(const int);
   void add_edge_loop(const int);
   void add_edge_loop_to_bound(const int, const int);
   void add_bound_to_face(const int, const int);
   int get_face_id(const int);
   int get_bound_id(const int);
   int get_edge_loop_id(const int);
   int get_edge_id(const int);
   void add_edge(const int);
   void add_edge_to_loop(const int, const int);
   void get_face(const int);
   void add_shell(const int);
   void create_face_objects(int);
   void print_no_of_faces();
   void print_no_of_edges();
   int return_no_of_faces();
   void print_edge_adjacent_faces(const int);
   void collect_edges();
   void find_all_model_edges();
   void print_list_of_edges();
   void print_list_of_bounds();
   void fetch_edge(const int, const int);
   void add_edge_relations();
   void print_face_bounds(const int);
   void print_bounds_containing_edges();
   void find_holes();
   void find_stiffeners();
   void update_adj_matrix();
   void print_adj_matrix(const int);
   int count_number_of_times_edge_used_in_loop(const int, const int);
   void generate_step_overlay();
   void add_vertex(const int, const float, const float, const float);
   void add_face_geometry_to_face(const int, const int);
   void add_vertex_to_face(const int, const int);
   int get_vertex_id(const int);
```

```
   void set_planar_face(const int);
   void set_cylindrical_face(const int);
   void set_face_normal(const int, const float, const float, const float);
   float get_face_angles(const int, const int);
   void find_2nd_order_junctions();
   void add_face_ideas_id(const std::string, const int);
   void create_feat_file();
   void output_high_order_edges();

private:
   Face facename[80];
   Vertex vertname[1400];
   char model_name[25];
   Edge edgename[250];
   Bound boundname[80];
   Edge_Loop loopname[80];
   int shell_label;
   int no_of_bounds;
   int no_of_faces;
   int no_of_loops;
   int no_of_face_inner_bounds;
   int no_of_edges;
   int rowsize;
   int colsize;
   int depth;
    vector<vector<vector<int> > > adjacency_matrix;
// Vectors are sized using resize function in main program
   int vecsize;
   vector<int> Edge_Orders;
   vector<int> Face_Bounds;
   vector<int> Rib_Faces;
   vector<int> Buttress_Faces;
   vector<int> Hole_Loops;
   vector<int> X_edges;
   vector<int> Very_high_edges;
   vector<int> Face_Features;
   vector<int> Surrounded_Faces;
   vector<int> Boss_Faces;
   vector<int> Fin_Faces;

   int no_of_buttresses;
   int no_of_ribs;
   int no_of_holes;
   int no_of_x_edges;
   int no_of_very_high_edges;
   int no_of_face_features;
   int no_of_surrounded_faces;
   int no_of_bosses;
   int no_of_fins;
   int no_of_vertices;
   int no_of_features;

};

#endif
```

```
/*
 * midsurf.cc
 *
 * Helen Lockett, Cranfield University
 *
 * June  2005
 *
 * h.lockett@cranfield.ac.uk
 *
 * Developed using NIST STEP Class Library
 *
 * Available for download from
 *
 * http://www.mel.nist.gov/msidstaff/sauder/SCL.htm#download
 *
 * Some sections adapted from SCL basic examples
 *
 * In particular treg.cc, Written by Ian Soboroff, NIST. June, 1994
 *
 */

#include <iostream>
#include <fstream>
#include <iomanip>
#include "midsurf.h"
#include <vector>

using namespace std;

/*---------------------------------------------------------------
 *      VERTEX
 *---------------------------------------------------------------
*/
Vertex::Vertex( )
{
}

/*---------------------------------------------------------------
 *      Vertex::add_vertex(const int) - Adds a vertex to the model
 *---------------------------------------------------------------
*/

void Vertex::add_vertex(const int vertexno, const float x_val, const float y_val, const
float z_val)
{
     coords.resize(3);
     vertex_id = vertexno;
     coords[0] = x_val;
     coords[1] = y_val;
     coords[2] = z_val;
}

/*---------------------------------------------------------------
 *      Vertex::get_vertex_id() - Gets the STEP ID for a vertex
 *---------------------------------------------------------------
*/

int Vertex::get_vertex_id()
{
    return vertex_id;
}
/*---------------------------------------------------------------
 *      Vertex::get_coord_component(const int component) - Gets a coordinate value for a
VERTEX
 *---------------------------------------------------------------
*/

float Vertex::get_coord_component(const int component)
{
   return coords[component];
}

/*---------------------------------------------------------------
 *      EDGE
 *---------------------------------------------------------------
```

```
*/

Edge::Edge( )
{
  no_of_faces_using_edge= 0;
  no_of_loops = 0;
}

/*-------------------------------------------------------------
 *      Edge::add_edge(const int) - Adds an edge to the model
 *-------------------------------------------------------------
*/

void Edge::add_edge(const int edgeno)
{
     edge_label = edgeno;
}


/*------------------------------------------------------------
 *       void Edge::add_loop_to_edge(const in loop_id)   - Adds an edge loop to an edge

 *------------------------------------------------------------
*/

void Edge::add_loop_to_edge(const int loop_id)
{
        no_of_loops = no_of_loops + 1;
        list_of_loops[no_of_loops] = loop_id;
}


/*-----------------------------------------------------------
 *         int Edge::get_edge_id() - Returns STEP ID for an edge

 *-----------------------------------------------------------
 */

int Edge::get_edge_id()
{
        return edge_label;
}


/*-----------------------------------------------------------
 *         int Edge::get_no_of_loops() - finds how many loops are connected to edge
 *-----------------------------------------------------------
 */

int Edge::get_no_of_loops()
{
        return no_of_loops;
}

/*-----------------------------------------------------------
 *         void Edge::add_face_to_edge(int) - add connected face to edge
 *
 *-----------------------------------------------------------
 */

void Edge::add_face_to_edge(int face)
{
    faces_using_edge[no_of_faces_using_edge] = face;
    no_of_faces_using_edge = no_of_faces_using_edge + 1;
}

/*-----------------------------------------------------------
 *         int Edge::get_no_of_faces_using_edge() - Returns the number of faces
connected to edge
 *-----------------------------------------------------------
 */

int Edge::get_no_of_faces_using_edge()
{
    return no_of_faces_using_edge;
}
```

```
/*-----------------------------------------------------------
 *         int Edge::get_face_id(const int faceid) - get the id of a face connected to
edge
 *-----------------------------------------------------------
 */

int Edge::get_face_id(const int faceid)
{
    return faces_using_edge[faceid];
}

/*-------------------------------------------------------------
 *      EDGE_LOOP
 *-------------------------------------------------------------
*/
Edge_Loop::Edge_Loop( )
{
        no_of_edges = 0;
        no_of_attached_faces = 0;
  }


/*-----------------------------------------------------------
 *         void Edge_Loop::add_edge_loop(const int loopno) - adds an edge_loop to the
model
 *-----------------------------------------------------------
 */

void Edge_Loop::add_edge_loop(const int loopno)
{
        edge_loop_label = loopno;
}

/*-----------------------------------------------------------
 *         void Edge_Loop::add_edge_to_loop(int edge_id) - adds an edge to a loop
 *-----------------------------------------------------------
 */
void Edge_Loop::add_edge_to_loop(const int edge_id)
{
        no_of_edges= no_of_edges + 1;
        list_of_edges[no_of_edges] = edge_id;
}


/*-----------------------------------------------------------
 *         int Edge_Loop::get_no_of_edges() - returns no of edges in loop
 *-----------------------------------------------------------
 */

int Edge_Loop::get_no_of_edges()
{
    return no_of_edges;
}


/*-----------------------------------------------------------
 *         int Edge_Loop::get_edge(const int index) - returns edge ID for edge in edge
loop
 *-----------------------------------------------------------
 */

int Edge_Loop::get_edge(const int index)
{
    return list_of_edges[index];

}
/*-----------------------------------------------------------
 *         void Edge_Loop::get_edge_loop_id() - returns STEP if for edge loop
 *-----------------------------------------------------------
 */

int Edge_Loop::get_edge_loop_id()
{
    return edge_loop_label;
}
```

```
/*--------------------------------------------------------------------
 *         void Edge_Loop::add_attached_face(const int) - Adds attached face to edge loop.
 *---------------------------------------------------------------------*/

void Edge_Loop::add_attached_face(const int att_fac)
   {
    no_of_attached_faces = no_of_attached_faces + 1;
     attached_faces[no_of_attached_faces] = att_fac;
   }


/*-----------------------------------------------------------------------
 *         int Edge_Loop::get_attached_face(const int att_fac)  - Returns STEP ID of
attached face
 *---------------------------------------------------------------------*/

int Edge_Loop::get_attached_face(const int att_fac)
   {
     return attached_faces[att_fac];
   }

/*------------------------------------------------------------------------
 *        int Edge_Loop::get_no_of_attached_faces()  - returns number of faces attached
to edge loop
 *------------------------------------------------------------------------*/

int Edge_Loop::get_no_of_attached_faces()
   {
     return no_of_attached_faces;
   }

/*-------------------------------------------------------------
 *      BOUND
 *-------------------------------------------------------------
*/

Bound::Bound( )
{
}


/*-----------------------------------------------------------
 *          void Bound::add_bound(const int boundno) - adds a bound to the model
 *-----------------------------------------------------------
 */

void Bound::add_bound(const int boundno)
{
     bound_label = boundno;
}

/*-----------------------------------------------------------
 *          void Bound::add_loop_to_bound(int loop_id) - adds an edge loop to the bound
 *-----------------------------------------------------------
 */

void Bound::add_loop_to_bound(const int loop_id)
{
     edge_loop_name = loop_id;
}


/*-----------------------------------------------------------
 *          void Bound::get_bound_id() - Returns STEP id  of bound
 *-----------------------------------------------------------
 */

int Bound::get_bound_id()
{
    return bound_label;
}

/*-----------------------------------------------------------
 *          void Bound::get_edge_loop_name() -    returns the edge loop ID for edge loop
of bound
```

```
 *----------------------------------------------------------
 */

int Bound::get_edge_loop_name()
{
    return edge_loop_name;
}


/*-------------------------------------------------------------
 *      FACE
 *-------------------------------------------------------------
*/

Face::Face( )
{
        no_of_edges = 0;
        face_label = 0;
        vertices.resize(25);
        no_of_vertices_in_face = 0;
        planar= 0;
        cylindrical = 0;
        normal_dir.resize(3);
}


/*-----------------------------------------------------------
 *         void Face::add_face(int facno) - Adds a face to the model
 *-----------------------------------------------------------
 */

void Face::add_face(int facno)
{
     face_label = facno;
}

/*-----------------------------------------------------------
 *         void Face::add_bound(int boundno) - adds a face bound to the model
 *-----------------------------------------------------------
 */

void Face::add_bound(int boundno)
{
        bound_label = boundno;
        no_of_bounds = no_of_bounds + 1;
        list_of_bounds[no_of_bounds] = boundno;
}


/*-----------------------------------------------------------
 *         void Face::get_bound_no_from_id(const int loopno) - returns STEP ID for bound
 *-----------------------------------------------------------
 */

int Face::get_bound_no_from_id(const int loopno)
{
   return list_of_bounds[loopno];
}

/*-----------------------------------------------------------
 *         void Face::get_bound_id(const int loopno) - finds a loop number in a bound
 *-----------------------------------------------------------
 */
int Face::get_bound_id(const int loopno)
{
        for(int counter = 0; counter <= no_of_bounds; counter++)
        {
                if (loopno == bound[counter].get_bound_id())
                {
                        return counter;
                }
        }
 }


/*-----------------------------------------------------------
```

```
 *         void Face::get_bound(const int boundno) - gets loop number from step ID
 *-----------------------------------------------------------
 */
int Face::get_bound(const int boundno)
{
       return bound[boundno].get_bound_id();
}


/*-----------------------------------------------------------
 *         int Face::get_face_id() - returns the STEP ID for a face
 *-----------------------------------------------------------
 */

int Face::get_face_id()
{
        return face_label;
}


/*-----------------------------------------------------------
 *         int Face::get_no_of_bounds()   -
 *-----------------------------------------------------------
 */

int Face::get_no_of_bounds()
{
    return no_of_bounds;
}


/*-----------------------------------------------------------
 *         void Face::add_face_geometry(const int face_geom_id) - add the STEP ID for
the face geom (surface)
 *-----------------------------------------------------------
*/

void Face::add_face_geometry(const int face_geom_id)

{
        face_geometry = face_geom_id;

}

/*-----------------------------------------------------------
 *         int Face::get_surface_geom_id() - returns the STEP ID of the face surface
 *-----------------------------------------------------------
*/
int Face::get_surface_geom_id()
{
        return face_geometry;
}

/*-----------------------------------------------------------
 *         void Face::add_vertex_to_face(const int vertex_id) - Adds a vertex to a face
 *-----------------------------------------------------------
*/

void Face::add_vertex_to_face(const int vertex_id)
{
  vertices[no_of_vertices_in_face] = vertex_id;
  no_of_vertices_in_face = no_of_vertices_in_face + 1;
}

/*-----------------------------------------------------------
 *         void Face::set_planar() - specifies a face as planar
 *-----------------------------------------------------------
*/

void Face::set_planar()
{
  planar = 1;
}

/*-----------------------------------------------------------
 *         void Face::set_cylindrical() - specifies a face as cylindrical
 *-----------------------------------------------------------
*/

void Face::set_cylindrical()
{
  cylindrical = 1;
}

 /*-----------------------------------------------------------
 *         void Face::get_planar() - returns value of planar
 *-----------------------------------------------------------
*/

int Face::get_planar()
{
  return planar;
}

 /*-----------------------------------------------------------
 *         void Face::get_cylindrical() - returns value of planar
 *-----------------------------------------------------------
*/

int Face::get_cylindrical()
{
        return cylindrical;
}

/*-----------------------------------------------------------
 *         int Face::get_vertex_id(const int vert_id) - returns the STEP ID of a vertex
 *-----------------------------------------------------------
*/
int Face::get_vertex_id(const int vert_id)
{
        return vertices[vert_id];
}

/*-----------------------------------------------------------
 *         void Face::set_normal_dir(const float xdir, const float ydir, const float
zdir) - defines normal directions for planar face
 *-----------------------------------------------------------
*/
        void Face::set_normal_dir(const float xdir, const float ydir, const float zdir)
{
 normal_dir[0] = xdir;
 normal_dir[1] = ydir;
 normal_dir[2] = zdir;

}

/*-----------------------------------------------------------
 *         float Face::get_normal_dir_component(const int dir) - returns  the normal
directions for a face
 *-----------------------------------------------------------
*/
float Face::get_normal_dir_component(const int dir)
{
        return normal_dir[dir];
}

/*-----------------------------------------------------------
 *         void Face::set_ideas_id(const std::string val) - sets the I-DEAS ID for a face
 *-----------------------------------------------------------
*/

void Face::set_ideas_id(const std::string val)
{
 ideas_id = val;
}

/*-----------------------------------------------------------
 *         std::string Face::get_ideas_id() - returns the value of the IDEAS ID for a
face
 *-----------------------------------------------------------
*/
```

```
std::string Face::get_ideas_id()
{
  return ideas_id;
}

/*---------------------------------------------------------------
 *      MIDSURFACE
 *---------------------------------------------------------------
*/
MidSurface::MidSurface( )
{
        no_of_faces = 0;
        no_of_edges = 0;
        no_of_vertices  = 0;
        no_of_features = 0;
        vecsize = 100;
        Edge_Orders.resize(2*vecsize);
        Face_Bounds.resize(vecsize);
        Rib_Faces.resize(vecsize);
        Buttress_Faces.resize(vecsize);
        Hole_Loops.resize(vecsize);
        Boss_Faces.resize(vecsize);
        Fin_Faces.resize(vecsize);
        X_edges.resize(vecsize);
        Very_high_edges.resize(vecsize);
        Face_Features.resize(vecsize);
        Surrounded_Faces.resize(vecsize);
}

/*-------------------------------------------------------------
 *          void MidSurface::add_face(const int face_id)  - Adds a face to the mid-
surface model
 *-------------------------------------------------------------
*/
void MidSurface::add_face(const int face_id)
{
        no_of_faces = no_of_faces + 1;
        facename[no_of_faces].add_face(face_id);

}

/*-------------------------------------------------------------
 *           void MidSurface::add_vertex(const int vertex_id, const float xval, const
float yval, const float zval) - Adds a Vertex
 *-------------------------------------------------------------
*/

void MidSurface::add_vertex(const int vertex_id, const float xval, const float yval,
const float zval)
{
        no_of_vertices = no_of_vertices + 1;
        vertname[no_of_vertices].add_vertex(vertex_id, xval, yval, zval);
}

/*-------------------------------------------------------------
 *           void MidSurface::add_face_geometry_to_face(const int face_id, const int
face_geom_id) - Adds a surface
 *-------------------------------------------------------------
*/

void MidSurface::add_face_geometry_to_face(const int face_id, const int face_geom_id)
{
        int face_no = get_face_id(face_id);
        facename[face_no].add_face_geometry(face_geom_id);
//      cout << "Adding face_geometry " << face_geom_id << " to face " << face_id <<
endl;
}

/*-------------------------------------------------------------
 *               int MidSurface::get_vertex_id(const int vertex_no) - returns STEP ID
for vertex
 *-------------------------------------------------------------
*/

int MidSurface::get_vertex_id(const int vertex_no)
{
        for (int counter = 0; counter <= no_of_vertices; counter++ )
        {
                if ( vertex_no == vertname[counter].get_vertex_id())
                return counter;
        }
}

/*-----------------------------------------------------------
 *               void MidSurface::add_vertex_to_face(const int face_id, const int
vertex_id) - associates vertex to a face
 *-----------------------------------------------------------
*/
void MidSurface::add_vertex_to_face(const int face_id, const int vertex_id)
{
        int face_no = get_face_id(face_id);
        int vertex_no= get_vertex_id(vertex_id);
//      std::cout << "vertex " << vertex_id << "added to face " << face_id << endl;
        facename[face_no].add_vertex_to_face(vertex_no);
//      std::cout << "Added Vertex " << vertex_no << " to face " << face_no << endl;
}

/*-----------------------------------------------------------
 *                 void MidSurface::set_planar_face(const int face_id)  - defines a face
as planar
 *-----------------------------------------------------------
*/

void MidSurface::set_planar_face(const int face_id)
{
        int face_no = get_face_id(face_id);
        facename[face_no].set_planar();
//      std::cout << "Face "<< face_id<< " is planar " << endl;
}

/*-----------------------------------------------------------
 *                   void MidSurface::set_cylindrical_face(const int face_id) - defines
a face as cylindrical
 *-----------------------------------------------------------
*/

void MidSurface::set_cylindrical_face(const int face_id)
{
        int face_no = get_face_id(face_id);
        facename[face_no].set_cylindrical();
//      cout << "Face "<< face_id<< " is cylindrical " << endl;
}

/*-----------------------------------------------------------
 *                    void MidSurface::set_face_normal(const int face_id, const float
xdir, const float ydir, const float zdir) - defines face normal direction
 *-----------------------------------------------------------
*/

void MidSurface::set_face_normal(const int face_id, const float xdir, const float ydir,
const float zdir)
{

        int face_no = get_face_id(face_id);
        facename[face_no].set_normal_dir(xdir, ydir, zdir);
}

/*-----------------------------------------------------------
 *               void MidSurface::add_shell(const int shell_id) - adds a shell
 *-----------------------------------------------------------
*/

void MidSurface::add_shell(const int shell_id)
{
        shell_label = shell_id;
//      std::cout << "Shell added with id " << shell_label << "\n";
}

/*-----------------------------------------------------------
 *               void MidSurface::add_bound(const int bound_id)  - adds a bound
 *-----------------------------------------------------------
*/
```

```

void MidSurface::add_bound(const int bound_id)
{
        no_of_bounds = no_of_bounds + 1;
        boundname[no_of_bounds].add_bound(bound_id);
//      std::cout << "no_of_bounds = " << no_of_bounds << "\n";
}

/*-----------------------------------------------------------
 *                void MidSurface::add_edge_loop(const int loop_id) - adds an edge loop
 *-----------------------------------------------------------
*/

void MidSurface::add_edge_loop(const int loop_id)
{
        no_of_loops = no_of_loops + 1;
        loopname[no_of_loops].add_edge_loop(loop_id);
//      std::cout << "no_of_bounds = " << no_of_bounds << "\n";
}

/*-----------------------------------------------------------
 *                void MidSurface::add_edge_loop_to_bound(const int bound_id, const int
loop_id) - associates edge loop to bound
 *-----------------------------------------------------------
*/

void MidSurface::add_edge_loop_to_bound(const int bound_id, const int loop_id)
{
        int bound_no = get_bound_id(bound_id);
//      std::cout << "The bound ID for " << bound_id << " is " << bound_no << "\n";
        boundname[bound_no].add_loop_to_bound(loop_id);
//      std::cout << "Adding loop " << loop_id << " to bound " << bound_no <<  "\n";

}

/*-----------------------------------------------------------
 *                void MidSurface::add_bound_to_face(const int face_id, const int
bound_id) - associated bound to face
 *-----------------------------------------------------------
*/

void MidSurface::add_bound_to_face(const int face_id, const int bound_id)
{
        int face_no = get_face_id(face_id);
        int bound_no = get_bound_id(bound_id);
//      std::cout << "The face ID for " << face_id << " is " << face_no << "\n";
        facename[face_no].add_bound(bound_no);
//      std::cout << "Adding Face bound " << bound_no << " to face " << face_no <<  "\n";
}

/*-----------------------------------------------------------
 *                 void MidSurface::add_edge(const int edge_id) - adds an edge
 *-----------------------------------------------------------
*/

void MidSurface::add_edge(const int edge_id)
{
        no_of_edges = no_of_edges + 1;
        edgename[no_of_edges].add_edge(edge_id);
//      std::cout << "no_of_edges = " << no_of_edges << "\n";
}

/*-----------------------------------------------------------
 *                  int MidSurface::get_face_id(const int facno)  - returns STEP ID for
a face
 *-----------------------------------------------------------
*/

int MidSurface::get_face_id(const int facno)
{
        for (int counter = 0; counter <= no_of_faces; counter++ )
        {
                if ( facno == facename[counter].get_face_id())
                return counter;
        }
}
```

```
/*-----------------------------------------------------------
 *                 int MidSurface::get_bound_id(const int boundno) - returns STEP ID for
a bound
 *-----------------------------------------------------------
*/

 int MidSurface::get_bound_id(const int boundno)
{
        for (int counter = 0; counter <= no_of_bounds; counter++ )
        {
                if ( boundno == boundname[counter].get_bound_id())
                return counter;
        }
}

/*-----------------------------------------------------------
 *                int MidSurface::get_edge_loop_id(const int loopno)  - returns STEP ID
for an edgelloop
 *-----------------------------------------------------------
*/

 int MidSurface::get_edge_loop_id(const int loopno)
{
        for (int counter = 0; counter <= no_of_loops; counter++ )
        {
                if ( loopno == loopname[counter].get_edge_loop_id())
                return counter;
        }
}

/*-----------------------------------------------------------
 *                 int MidSurface::get_edge_id(const int edgeno) - returns STEP ID for
an edge
 *-----------------------------------------------------------
*/

 int MidSurface::get_edge_id(const int edgeno)
{
        for (int counter = 0; counter <= no_of_edges; counter++ )
        {
                if ( edgeno == edgename[counter].get_edge_id())
                return counter;
        }
}

/*-----------------------------------------------------------
 *                   void MidSurface::add_edge_to_loop(const int loop_id, const int
edge_id)  - associates an edge to an edge loop
 *-----------------------------------------------------------
*/

void MidSurface::add_edge_to_loop(const int loop_id, const int edge_id)
{
        int loop_no = get_edge_loop_id(loop_id);
        int edge_no = get_edge_id(edge_id);
        loopname[loop_no].add_edge_to_loop(edge_id);
        edgename[edge_no].add_loop_to_edge(loop_id);
 }

 /*-----------------------------------------------------------
 *                 int MidSurface::return_no_of_faces() - returns number of faces in
model
 *-----------------------------------------------------------
*/

int MidSurface::return_no_of_faces()
{
        return no_of_faces;
}

/*-----------------------------------------------------------
 *                void MidSurface::print_no_of_faces() - prints number of faces
 *-----------------------------------------------------------
*/
```

```
void MidSurface::print_no_of_faces()
{
        std::cout << "The number of faces is " << no_of_faces << "\n";
}

/*-----------------------------------------------------------
 *                 void MidSurface::print_no_of_edges() - prints number of edges
 *-----------------------------------------------------------
*/

void MidSurface::print_no_of_edges()
{
        std::cout << "The number of edges is " << no_of_edges << "\n";
}


/*-----------------------------------------------------------
 *                 void MidSurface::print_list_of_edges()  - prints a list of the edges
in model
 *-----------------------------------------------------------
*/

void MidSurface::print_list_of_edges()
{
        std::cout << "In printing routine" << "\n";
        for (int count = 1; count <= no_of_edges; count++)
        {
                std::cout << "Edge number " << edgename[count].get_edge_id() << "\n";
        }
}

/*-----------------------------------------------------------
 *                  void MidSurface::print_list_of_bounds() - prints a list of bounds in
model
 *-----------------------------------------------------------
*/

void MidSurface::print_list_of_bounds()
{
        for (int counter = 1; counter <= no_of_bounds; counter++)
       {
                int tmpboundno = boundname[counter].get_bound_id();
                std::cout << "Bound number " << counter << "  = " << tmpboundno << "\n";
       }
}

/*-----------------------------------------------------------
 *                   void MidSurface::print_face_bounds(const int face_id) - prins list
of bounds in face
 *-----------------------------------------------------------
*/

void MidSurface::print_face_bounds(const int face_id)
{
     int face_no = get_face_id(face_id);
     std::cout <<"Face ID = " << face_no << "\n";
}

 /*-----------------------------------------------------------
 *                   void MidSurface::print_bounds_containing_edges()  - prints all
bounds using an edge
 *-----------------------------------------------------------
*/

void MidSurface::print_bounds_containing_edges()
{
        for (int ecount = 1; ecount <=no_of_edges; ecount++)
        {
                std::cout << "Edge " << ecount;
        }
}

 /*-----------------------------------------------------------
 *         int MidSurface::count_number_of_times_edge_used_in_loop(const int, const int)
  -   Find reused edges in loop
 *-----------------------------------------------------------
 */

int MidSurface::count_number_of_times_edge_used_in_loop(const int edgeno, const int
loopno)
{
        int loop_id = get_edge_loop_id(loopno);
        int no_of_edges_in_loop = loopname[loop_id].get_no_of_edges();
        int no_of_instances_of_edge = 0;
        for (int count= 1; count <= no_of_edges_in_loop; count++)
        {
                int current_edge = loopname[loop_id].get_edge(count);
                if (current_edge == edgeno)
                {
                        no_of_instances_of_edge =  no_of_instances_of_edge +1;
                }
        }
        return no_of_instances_of_edge;
}

 /*-----------------------------------------------------------
 *                 void MidSurface::add_face_ideas_id(const std::string ideasid, const
int face_no) - adds I-DEAS IDto a face
 *-----------------------------------------------------------
*/

void MidSurface::add_face_ideas_id(const std::string ideasid, const int face_no)
{
        int face_id = get_face_id(face_no);
        facename[face_id].set_ideas_id(ideasid);
}

/*-----------------------------------------------------------
 *                 void MidSurface::update_adj_matrix() - populates the adjacency matrix
 *-----------------------------------------------------------
*/

void MidSurface::update_adj_matrix()
{
// Take care that rowsize is edges, colsize is attributes and depth is faces
        rowsize= 400;
        colsize= 3;
        depth = 200;
        vector<int> Row(colsize,0);
        vector <vector<int> > Table(rowsize, Row);
        vector <vector< vector <int> > > adj_matrix(depth, Table);

        for (int a = 1; a <= no_of_faces; a++)
        {
                for (int b = 1; b <= no_of_edges; b++)
                {
                        adj_matrix [a][b][1] = 0;
                        adj_matrix [a][b][2] = 0;
                        adj_matrix [a][b][3] = 0;
                }
        }

        std::cout << "Populating adjacency matrix"<< "\n";
        std::cout << "Number of faces is " << no_of_faces << "\n";
        std::cout << "Number of edges is " << no_of_edges << "\n";
        for (int i = 1; i <= no_of_faces; i++)
        {
                int no_of_bounds_in_face = facename[i].get_no_of_bounds();
                for (int j = 1; j <= no_of_bounds_in_face; j++)
                {
                        int current_bound = facename[i].get_bound_no_from_id(j);
                        int current_loop = boundname[current_bound].get_edge_loop_name();
                        int current_loop_id = get_edge_loop_id(current_loop);
                        int no_of_edges_in_loop =
loopname[current_loop_id].get_no_of_edges();

                        for (int k = 1; k <= no_of_edges_in_loop; k++)
                        {
                                int current_edge_id =
loopname[current_loop_id].get_edge(k);
                                int current_edge_label = get_edge_id(current_edge_id);
                                //Write array elements
```

```
                                //Loop number
                                adj_matrix [i][current_edge_label][0] = current_loop;
                                //Bound Value in Face
                                adj_matrix [i][current_edge_label][1] = j;
                                //Number of edge uses in loop
                                adj_matrix [i][current_edge_label][2] =
count_number_of_times_edge_used_in_loop(current_edge_id, current_loop);
                        }
                }
        }

adjacency_matrix = adj_matrix;

//Populate the array Edge_Orders
// that contains the order of each edge in the model
// note that it takes into account the REUSED EDGES as well
//as the number of different faces incident on an edge
//
for ( int j=1; j <= no_of_edges; j++)
        {
        Edge_Orders[j] = 0;
        for ( int i = 1; i <= no_of_faces; i++)
                {
                        Edge_Orders[j] = Edge_Orders[j] + adjacency_matrix[i][j][2];
                        if (adjacency_matrix[i][j][2] != 0)
                        {
                           edgename[j].add_face_to_edge(i);
                        }
                }
        }

// Routine to update facebounds in adjacency matrix

for (int i=1; i <= no_of_faces; i++)
        {
                Face_Bounds[i] = 0;
                for (int j = 1; j <= no_of_edges; j++)
                        {
                                if (adjacency_matrix[i][j][1] > Face_Bounds[i])
                                        Face_Bounds[i] = adjacency_matrix[i][j][1];
                        }
        }

}

/*-----------------------------------------------------------
 *                  void MidSurface::print_adj_matrix(const int value)   - Print
adjacency matrix to screen
 *-----------------------------------------------------------
*/

void MidSurface::print_adj_matrix(const int value)
{
        std::cout << "Printing Adjacency matrix" << "\n" << "\n" ;
        std::cout << "Matrix shows adjacency between model faces and edges" << "\n";
        std::cout << "Matrix values are the edge loop membership for edge on face" <<
"\n";

        std::cout << "\t";
for (int h = 1; h<=no_of_edges; h++)
        {
                std::cout << edgename[h].get_edge_id() << "\t";
        }
        std::cout << "\n";

for (int i = 1; i <= no_of_faces; i++)
        {
        std::cout << facename[i].get_face_id() << "\t";

                for (int j = 1; j <= no_of_edges; j++)
                {
                        std::cout << adjacency_matrix[i][j][value] << "\t";
                }
                std::cout << Face_Bounds[i] << "\n";
        }
```

```
std::cout << "\t";
for (int h = 1; h <= no_of_edges; h++)
        {
                std::cout  << Edge_Orders[h] << "\t";
        }
        std::cout << "\n";
}

/*------------------------------------------------------------
 *               void MidSurface::find_holes()   - Routine to fine face features
 *-----------------------------------------------------------
*/

void MidSurface::find_holes()
{
        ofstream FeatFile("feat-file.txt", ios::app);
        no_of_features = 0;
        no_of_holes = 0;
        no_of_face_features = 0;
        no_of_bosses = 0;
        no_of_fins = 0;
//for each face in model
        for (int counter = 1; counter <= no_of_faces; counter++)
        {
//          cout << "Number of features is " << no_of_features << endl;
// if face has two or more bounds
                if (Face_Bounds[counter] >= 2)
                {
//              std::cout << "The current face no is "  << counter << endl;
// for each loop in face (except outer bound)
                        for (int loopcounter = 2; loopcounter <= Face_Bounds[counter];
loopcounter++)
                        {
//get the id for the current bound
                                int current_bound_no =
facename[counter].get_bound_no_from_id(loopcounter);
//
//Find out how many edges are in the loop
//
                                int no_of_edges_in_loop =
loopname[current_bound_no].get_no_of_edges();
//                      cout << " no_of_edges_in_loop is " << no_of_edges_in_loop <<
"\n";
//For each edge
                                for (int edgecount = 1; edgecount <=
no_of_edges_in_loop; edgecount++)
                                {
                                        int model_edge_no =
loopname[current_bound_no].get_edge(edgecount);
//                      cout << "value of model_edge_no  is " << model_edge_no << "\n";
                                        int model_edge_id = get_edge_id(model_edge_no);
//                      cout <<"value of model_edge_id is " << model_edge_id << "\n";
//                      cout << " Value of Edge_Orders[model_edge_id] is " <<
Edge_Orders[model_edge_id] << "\n";
                                        if (Edge_Orders[model_edge_id] > 1 )
                                        {
                                                for (int facecounter = 1; facecounter
<=no_of_faces; facecounter++)
                                                {
                                                        if
(adjacency_matrix[facecounter][model_edge_id][1]> 0 && counter != facecounter &&
facename[facecounter].get_face_id() != Face_Features[no_of_face_features])
                                                        {
//                                     no_of_attached_faces = no_of_attached_faces + 1;

loopname[current_bound_no].add_attached_face(facecounter);
                                                                no_of_face_features =
no_of_face_features + 1;
//                                                                 std::cout << "Adding
attached face " << facecounter << "to face " << counter << endl;


Face_Features[no_of_face_features]= facename[facecounter].get_face_id();
//                              std::cout << "value of Face_features is " <<
Face_Features[no_of_face_features] << endl;
                                                        }
```

```
                                                }
                                        }
                                }


                        int no_of_attached_faces =
loopname[current_bound_no].get_no_of_attached_faces();
//                        cout << "No. of attached faces for loop " << current_bound_no
<< " on face " << facename[counter].get_face_id() << " is " << no_of_attached_faces <<
"\n";
                        if (no_of_attached_faces == 0 )
                        {
                                no_of_holes = no_of_holes + 1;
                                Hole_Loops[no_of_holes] =
loopname[current_bound_no].get_edge_loop_id();
                                cout << "Loop " << Hole_Loops[no_of_holes] << " on Face
" << facename[counter].get_face_id() << " is a HOLE " << "\n";

                                no_of_features = no_of_features + 1;
                                FeatFile << "(feature (feat-number " << no_of_features
<< ") (feat-type hole) (has-thickness NO))" << endl;
                        }
                        else if (no_of_attached_faces >= 1)
                        {
//        cout << "Loop " << loopname[loopcounter].get_edge_loop_id() << " on Face " <<
facename[counter].get_face_id() << " connects to a FACE FEATURE containing face(s) " ;

                                for (int c= 1; c <= no_of_attached_faces; c++)
                                {
                                        int sub_face_no =
loopname[current_bound_no].get_attached_face(c);
                                        if (facename[sub_face_no].get_planar() == 1)
                                        {
                                          //test for angle between fin face and parent
face
                                                if  (facename[counter].get_planar() == 1)
                                                {
                                                        float angle_between =
get_face_angles(counter, sub_face_no);
                                                        cout << " Angle between faces is
" << angle_between << endl;
                                                        if (angle_between > 5.0)
                                                        {
                  //                                               cout << " Faces are
planar, non parallel " << angle_between << "degrees" << endl;
                                                                cout << "Face " <<
facename[sub_face_no].get_face_id() << " is a FIN with planar faces" << endl;
                                                                no_of_features =
no_of_features + 1;
                                                                no_of_fins = no_of_fins
+ 1;
                                                                Fin_Faces[no_of_fins] =
facename[sub_face_no].get_face_id();
                                                                FeatFile << "(feature
(feat-number " << no_of_features << ") (feat-type fin) (id \"" <<
facename[sub_face_no].get_ideas_id() << "  \"   ) (has-thickness YES) (thickness %))" <<
endl;
                                                        }
                                                }
                                                else
                                                {
                  //                                       cout << "Parent face is non-
planar" << endl;
                                                        cout << "Face " <<
facename[sub_face_no].get_face_id() << " is a FIN  with non-planar parent face" << endl;
                                                        no_of_features = no_of_features
+ 1;
                                                        no_of_fins = no_of_fins + 1;
                                                        Fin_Faces[no_of_fins] =
facename[sub_face_no].get_face_id();
                                                        FeatFile << "(feature (feat-
number " << no_of_features << ") (feat-type fin) (id \"" <<
facename[sub_face_no].get_ideas_id() << "  \"   ) (has-thickness YES) (thickness %))" <<
endl;
                                                }
                                                }
                                        else if
(facename[sub_face_no].get_cylindrical()== 1)
                                        {
                                                cout << "Face "<<
facename[sub_face_no].get_face_id() << " is a BOSS " << endl;
                                                no_of_features = no_of_features + 1;
                                                no_of_bosses = no_of_bosses + 1;
                                                Boss_Faces[no_of_bosses] =
facename[sub_face_no].get_face_id();
                                                FeatFile << "(feature (feat-number " <<
no_of_features << ") (feat-type boss) (id \"" << facename[sub_face_no].get_ideas_id() <<
"  \"   ) (has-thickness YES) (thickness %))" << endl;
                                        }
                                }
                        }
                }
        }
}
FeatFile.close();
}


/*---------------------------------------------------------
 *                    void MidSurface::find_stiffeners()  - Routine to find stiffener
features
 *---------------------------------------------------------
*/

void MidSurface::find_stiffeners()
{
        no_of_buttresses = 0;
        no_of_ribs = 0;
        no_of_surrounded_faces =0;
        ofstream FeatFile("feat-file.txt", ios::app);
        for (int facecounter=1; facecounter <= no_of_faces; facecounter++)
        {
          int loop_id = 0;
          for (int counter=1; counter <= no_of_edges; counter++)
          {
                   if (adjacency_matrix[facecounter][counter][1] == 1 )
                   {
                      loop_id = adjacency_matrix[facecounter][counter][0];
                      break;
                   }
          }
                  if (loop_id == 0 )
                  {
                    cout << " Error: Face " << facecounter << " does not have an outer
loop " << endl;
                  }

                  int loop_no = get_edge_loop_id(loop_id);
//                  cout << "Value of loop_no is " << loop_no << endl;
                  int no_of_edges_in_loop = loopname[loop_no].get_no_of_edges();
                  int list_of_edge_orders_for_loop[25] = {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0};
                                        for (int edgecounter = 1; edgecounter <=
no_of_edges_in_loop; edgecounter++)
                                        {
                                                int edge_no =
loopname[loop_no].get_edge(edgecounter);
                                                int edge_id = get_edge_id(edge_no);

                                             list_of_edge_orders_for_loop[edgecounter] =
Edge_Orders[edge_id];
                                        }
                                int no_of_adj_high_order_edges = 0;
                                int no_of_unconnected_edges = 0;
                                int max_no_of_adj_high_order_edges = 0;
                                for (int edgecheck = 1; edgecheck <=
no_of_edges_in_loop; edgecheck++)
                                {
                                        if (list_of_edge_orders_for_loop[edgecheck] == 1)
                                        {
                                           no_of_unconnected_edges =
no_of_unconnected_edges + 1;
```

```
                                        }
                                        if (list_of_edge_orders_for_loop[edgecheck] >=3)
                                        {
                                           no_of_adj_high_order_edges =
no_of_adj_high_order_edges + 1;
                                           max_no_of_adj_high_order_edges =
no_of_adj_high_order_edges;
                                        }
                                        else
                                        no_of_adj_high_order_edges = 0;
                                }
                                if (list_of_edge_orders_for_loop[no_of_edges_in_loop] >=
3 && list_of_edge_orders_for_loop[1] >=3 && no_of_edges_in_loop >=3)
                                {
                                        no_of_adj_high_order_edges =
no_of_adj_high_order_edges + 1;
                                        max_no_of_adj_high_order_edges =
no_of_adj_high_order_edges;
                                }
                        if (max_no_of_adj_high_order_edges == 2 && no_of_edges_in_loop >
2 && no_of_unconnected_edges > 0 && facename[facecounter].get_planar() == 1)
                        {
                                no_of_buttresses= no_of_buttresses + 1;
                                Buttress_Faces[no_of_buttresses] =
facename[facecounter].get_face_id();
                                cout << "Loop " << loop_id << " on Face " <<
Buttress_Faces[no_of_buttresses] << " is a BUTTRESS " << "\n";
                                no_of_features = no_of_features + 1;
                                FeatFile << "(feature (feat-number " << no_of_features
<< ") (feat-type buttress) (id \"" << facename[facecounter].get_ideas_id() << "  \"   )
(has-thickness YES) (thickness %))" << endl;
                       }
                        else if (max_no_of_adj_high_order_edges >= 3 &&
no_of_edges_in_loop >= (max_no_of_adj_high_order_edges + 1) && no_of_unconnected_edges >
0 && facename[facecounter].get_planar() == 1)
                        {
                                no_of_ribs= no_of_ribs + 1;
                                Rib_Faces[no_of_ribs] =
facename[facecounter].get_face_id();
                                cout << "Loop " << loop_id << " on Face " <<
Rib_Faces[no_of_ribs] << " is a RIB " << "\n";
                                no_of_features = no_of_features + 1;
                                FeatFile << "(feature (feat-number " << no_of_features
<< ") (feat-type rib) (id \"" << facename[facecounter].get_ideas_id() << "  \"   ) (has-
thickness YES) (thickness %))" << endl;
                       }

//Code to recognise surrounded faces - not used at present

//                        else if (no_of_unconnected_edges == 0)
//                        {
//                                no_of_surrounded_faces= no_of_surrounded_faces + 1;
//                                Surrounded_Faces[no_of_surrounded_faces] =
facename[facecounter].get_face_id();
//                                cout << "Loop " << loop_id <<" on Face "<<
Surrounded_Faces[no_of_surrounded_faces] << " is a SURROUNDED FACE " << "\n";
//                                ofstream FeatFile("feat-file.txt", ios::app);
//                                no_of_features = no_of_features + 1;
//                                FeatFile << "(feature (feat-number " << no_of_features
<< ") (feat-type main-wall) (ideas-id " << facename[facecounter].get_ideas_id() << "   )
(has-thickness YES) (thickness %))" << endl;
//                                FeatFile.close();
//                        }
                        else
                        {
                                int is_face_feat = 0;
                                for (int facfeat = 1; facfeat <=no_of_face_features;
facfeat++)
                                        {
                                          if (facename[facecounter].get_face_id() ==
Face_Features[facfeat])
                                                {
                                                        is_face_feat = 1;
                                                        break;
                                                }
                                        }
                                if (is_face_feat == 0)
                                {
                                        no_of_features = no_of_features + 1;
                                        FeatFile << "(feature (feat-number " <<
no_of_features << ") (feat-type main-wall) (id \"" <<
facename[facecounter].get_ideas_id() << "  \"   ) (has-thickness YES) (thickness %))" <<
endl;
                                }
                        }
                }
FeatFile.close();
}

/*-----------------------------------------------------------
 *                      void MidSurface::output_high_order_edges() - write high order
edges to feature file
 *-----------------------------------------------------------
*/

void MidSurface::output_high_order_edges()
{
ofstream FeatFile("feat-file.txt", ios::app);
for (int edgecount = 1; edgecount <=no_of_edges; edgecount++)
        {
          if (Edge_Orders[edgecount] >=3 )
                        {
                        no_of_features = no_of_features + 1;
                        FeatFile << "(feature (feat-number " << no_of_features << ")
(feat-type ho-edge) (id \"" << edgename[edgecount].get_edge_id() << "\"   ) (edge-order
" << Edge_Orders[edgecount] << " ))" << endl;
                        }
        }
FeatFile.close();
}

/*-----------------------------------------------------------
 *                     float MidSurface::get_face_angles(const int face1, const int
face2) - claculate angles between faces
 *-----------------------------------------------------------
*/

float MidSurface::get_face_angles(const int face1, const int face2)
{

        float theta;
        float costheta;
        float topline = (facename[face1].get_normal_dir_component(0)*
facename[face2].get_normal_dir_component(0)
             + facename[face1].get_normal_dir_component(1)*
facename[face2].get_normal_dir_component(1)
             + facename[face1].get_normal_dir_component(2)*
facename[face2].get_normal_dir_component(2));
        float bottomline = (sqrt((facename[face1].get_normal_dir_component(0)*
facename[face1].get_normal_dir_component(0))
               +  (facename[face1].get_normal_dir_component(1)*
facename[face1].get_normal_dir_component(1))
                 + (facename[face1].get_normal_dir_component(2)*
facename[face1].get_normal_dir_component(2)))
                 *
               sqrt((facename[face2].get_normal_dir_component(0)*
facename[face2].get_normal_dir_component(0))
               +  (facename[face2].get_normal_dir_component(1)*
facename[face2].get_normal_dir_component(1))
                 + (facename[face2].get_normal_dir_component(2)*
facename[face2].get_normal_dir_component(2))));

        costheta = topline / bottomline;
        theta = acos(costheta)/3.14159*180;
        return theta;
}

/*-----------------------------------------------------------
 *              void MidSurface::find_2nd_order_junctions() -   Function to idnetify 2nd
order junctions using angles
 *-----------------------------------------------------------
```

```
*/

void MidSurface::find_2nd_order_junctions()

{

for ( int j=1; j <= no_of_edges; j++)
        {
        if (Edge_Orders[j] == 2)
        {
        int faceone = edgename[j].get_face_id(0);
        int facetwo = edgename[j].get_face_id(1);
        int faces_planar =facename[faceone].get_planar() +
facename[facetwo].get_planar();
        if (faceone != 0 && facetwo != 0 && faces_planar == 2)
               {
               cout << "Along edge " << j << " The angle between faces " <<  faceone <<
" and " << facetwo
               << " is " << get_face_angles(edgename[j].get_face_id(0),
edgename[j].get_face_id(1)) << " degrees " <<  endl;
               }
        }
        }

}

/*-----------------------------------------------------------
 *         void MidSurface::generate_step_overlay()  - Routine to write the results as a
STEP overlay for display
 *-----------------------------------------------------------
*/

void MidSurface::generate_step_overlay()
//
// Function to assign colour coding to the features in the model, based on the feature
recognition results
// Generates a file called step-col.txtthat can be inserted into the original STEP file
// Needs a shell script to automate this process
//
// Note that the starting STEP ID is hard coded
//
{
        int start_at = 5000; //check for large files
        int counter = 0;
        int no_of_styled_items = 0;
        int blue_id = 0;
        int orange_id = 0;
        int green_id = 0;
        int red_id= 0;
        int yellow_id = 0;
        int cyan_id = 0;
        int styled_item_id = 0;
        int global_rep = 28;
        ofstream SaveFile("step-col.txt");
        counter = start_at;
        SaveFile << "#" << counter << "=COLOUR_RGB('LIGHT_BLUE',0.0,0.66,1.0);" << endl;
        counter = counter +1;
        SaveFile << "#" << counter <<
"=DRAUGHTING_PRE_DEFINED_CURVE_FONT('continuous');" << endl;
        counter = counter + 1;
        SaveFile << "#" << counter << "=CURVE_STYLE('',#" << counter - 1 << ",
POSITIVE_LENGTH_MEASURE(0.1), #" << counter - 2 << "); " << endl;
        counter = counter + 1;
        SaveFile << "#" << counter << "=FILL_AREA_STYLE_COLOUR('',#" << counter - 3 <<
");" << endl;
        counter = counter + 1;
        SaveFile << "#" << counter << "=FILL_AREA_STYLE('',(#" << counter - 1 << "));"
<< endl;
        counter = counter + 1;
        SaveFile << "#" << counter <<"=SURFACE_STYLE_FILL_AREA(#" << counter - 1 << ");"
<< endl;
        counter = counter + 1;
        SaveFile << "#" << counter << "=SURFACE_SIDE_STYLE('',(#" << counter - 1 <<
"));" << endl;
        counter = counter + 1;
        SaveFile << "#" << counter << "=SURFACE_STYLE_USAGE(.BOTH.,#" << counter -1 <<
");" << endl;
```

```
        counter = counter + 1;
        SaveFile << "#" << counter << "=PRESENTATION_STYLE_ASSIGNMENT((#" << counter - 6
<< ",#" << counter - 1 << "));" << endl;
        blue_id = counter;
        counter = counter + 1;
        SaveFile << "#" << counter << "=COLOUR_RGB('YELLOW',1.0,1.0,0.0);" << endl;
        counter = counter + 1;
        SaveFile << "#" << counter << "=CURVE_STYLE('',#" << counter - 9 << ",
POSITIVE_LENGTH_MEASURE(0.1), #" << counter - 1 << "); " << endl;
        counter = counter + 1;
        SaveFile << "#" << counter << "=FILL_AREA_STYLE_COLOUR('',#" << counter - 2 <<
");" << endl;
        counter = counter + 1;
        SaveFile << "#" << counter << "=FILL_AREA_STYLE('',(#" << counter - 1 << "));"
<< endl;
        counter = counter + 1;
        SaveFile << "#" << counter <<"=SURFACE_STYLE_FILL_AREA(#" << counter - 1 << ");"
<< endl;
        counter = counter + 1;
        SaveFile << "#" << counter << "=SURFACE_SIDE_STYLE('',(#" << counter - 1 <<
"));" << endl;
        counter = counter + 1;
        SaveFile << "#" << counter << "=SURFACE_STYLE_USAGE(.BOTH.,#" << counter - 1 <<
");" << endl;
        counter = counter + 1;
        SaveFile << "#" << counter << "=PRESENTATION_STYLE_ASSIGNMENT((#" << counter - 6
<< ",#" << counter - 1 << "));" << endl;
        yellow_id = counter;
        counter = counter + 1;
        SaveFile << "#" << counter << "=COLOUR_RGB('CYAN',0.0,1.0,1.0);" << endl;
        counter = counter + 1;
        SaveFile << "#" << counter << "=CURVE_STYLE('',#" << counter - 9 << ",
POSITIVE_LENGTH_MEASURE(0.1), #" << counter - 1 << "); " << endl;
        counter = counter + 1;
        SaveFile << "#" << counter << "=FILL_AREA_STYLE_COLOUR('',#" << counter - 2 <<
");" << endl;
        counter = counter + 1;
        SaveFile << "#" << counter << "=FILL_AREA_STYLE('',(#" << counter - 1 << "));"
<< endl;
        counter = counter + 1;
        SaveFile << "#" << counter <<"=SURFACE_STYLE_FILL_AREA(#" << counter - 1 << ");"
<< endl;
        counter = counter + 1;
        SaveFile << "#" << counter << "=SURFACE_SIDE_STYLE('',(#" << counter - 1 <<
"));" << endl;
        counter = counter + 1;
        SaveFile << "#" << counter << "=SURFACE_STYLE_USAGE(.BOTH.,#" << counter - 1 <<
");" << endl;
        counter = counter + 1;
        SaveFile << "#" << counter << "=PRESENTATION_STYLE_ASSIGNMENT((#" << counter - 6
<< ",#" << counter - 1 << "));" << endl;
        cyan_id = counter;
        counter = counter + 1;
        SaveFile << "#" << counter << "=COLOUR_RGB('GREEN',0.0,1.0,0.0);" << endl;
        counter = counter + 1;
        SaveFile << "#" << counter << "=CURVE_STYLE('',#" << counter - 17 << ",
POSITIVE_LENGTH_MEASURE(0.1), #" << counter - 1 << "); " << endl;
        counter = counter + 1;
        SaveFile << "#" << counter << "=FILL_AREA_STYLE_COLOUR('',#" << counter - 2 <<
");" << endl;
        counter = counter + 1;
        SaveFile << "#" << counter << "=FILL_AREA_STYLE('',(#" << counter - 1 << "));"
<< endl;
        counter = counter + 1;
        SaveFile << "#" << counter <<"=SURFACE_STYLE_FILL_AREA(#" << counter - 1 << ");"
<< endl;
        counter = counter + 1;
        SaveFile << "#" << counter << "=SURFACE_SIDE_STYLE('',(#" << counter - 1 <<
"));" << endl;
        counter = counter + 1;
        SaveFile << "#" << counter << "=SURFACE_STYLE_USAGE(.BOTH.,#" << counter - 1 <<
");" << endl;
        counter = counter + 1;
        SaveFile << "#" << counter << "=PRESENTATION_STYLE_ASSIGNMENT((#" << counter - 6
<< ",#" << counter - 1 << "));" << endl;
        green_id = counter;
```

```
        counter = counter + 1;
        SaveFile << "#" << counter << "=COLOUR_RGB('RED',1.0,0.0,0.0);" << endl;
        counter = counter + 1;
        SaveFile << "#" << counter << "=CURVE_STYLE('',#" << counter - 25 << ",
POSITIVE_LENGTH_MEASURE(0.1), #" << counter - 1 << "); " << endl;
        counter = counter + 1;
        SaveFile << "#" << counter << "=FILL_AREA_STYLE_COLOUR('',#" << counter - 2 <<
");" << endl;
        counter = counter + 1;
        SaveFile << "#" << counter << "=FILL_AREA_STYLE('',(#" << counter - 1 << "));"
<< endl;
        counter = counter + 1;
        SaveFile << "#" << counter <<"=SURFACE_STYLE_FILL_AREA(#" << counter - 1 << ");"
<< endl;
        counter = counter + 1;
        SaveFile << "#" << counter << "=SURFACE_SIDE_STYLE('',(#" << counter - 1 <<
"));" << endl;
        counter = counter + 1;
        SaveFile << "#" << counter << "=SURFACE_STYLE_USAGE(.BOTH.,#" << counter - 1 <<
");" << endl;
        counter = counter + 1;
        SaveFile << "#" << counter << "=PRESENTATION_STYLE_ASSIGNMENT((#" << counter - 6
<< ",#" << counter - 1 << "));" << endl;
        red_id = counter;
        counter = counter + 1;
        SaveFile << "#" << counter << "=STYLED_ITEM('',(#" << green_id << "),#" <<
shell_label << ");" << endl;
        styled_item_id = counter;
          counter = counter + 1;
          no_of_styled_items = 1;
        for (int facecount = 1; facecount <= no_of_ribs; facecount++)
        {
                SaveFile << "#" << counter << "=OVER_RIDING_STYLED_ITEM('',(#" <<
blue_id << "),#" << Rib_Faces[facecount] << ",#" << styled_item_id << ");" << endl;
                no_of_styled_items = no_of_styled_items + 1;
                counter = counter+ 1;
        }
         for (int facecount = 1; facecount <= no_of_buttresses; facecount++)
        {
                SaveFile << "#" << counter << "=OVER_RIDING_STYLED_ITEM('',(#" <<
green_id << "),#" << Buttress_Faces[facecount] << ",#" << styled_item_id << ");" << endl;
                no_of_styled_items = no_of_styled_items + 1;
                counter = counter+ 1;
        }
          for (int facecount = 1; facecount <= no_of_bosses; facecount++)
        {
                SaveFile << "#" << counter << "=OVER_RIDING_STYLED_ITEM('',(#" <<
yellow_id << "),#" << Boss_Faces[facecount] << ",#" << styled_item_id << ");" << endl;
                no_of_styled_items = no_of_styled_items + 1;
                counter = counter+ 1;
        }
          for (int facecount = 1; facecount <= no_of_fins; facecount++)
        {
                SaveFile << "#" << counter << "=OVER_RIDING_STYLED_ITEM('',(#" <<
cyan_id << "),#" << Fin_Faces[facecount] << ",#" << styled_item_id << ");" << endl;
                no_of_styled_items = no_of_styled_items + 1;
                counter = counter+ 1;
        }
        SaveFile << "#" << counter <<
"=MECHANICAL_DESIGN_GEOMETRIC_PRESENTATION_REPRESENTATION('',(";
        for (int stylecount = 0; stylecount < no_of_styled_items - 1; stylecount++)
            {
             SaveFile << "#"  << counter - no_of_styled_items + stylecount << ",";
            }
        SaveFile << "#" << counter - 1;
        SaveFile << "),#" << global_rep << ");" << endl;
        SaveFile.close();
}

/*-------------------------------------------------------------
 *                   void MidSurface::create_feat_file() - Create the feature file
 *------------------------------------------------------------
*/

void MidSurface::create_feat_file()
// Function to create the feature file for the CLIPS knowledge base to read
```

```
//
{
        ofstream FeatFile("feat-file.txt");
        FeatFile << "(feature-model (thickness 4.0))" << endl;
        FeatFile.close();
}
```

```
/*
 * geomfromstep.cc
 *
 * Helen Lockett, Cranfield University
 *
 * June  2005
 *
 * h.lockett@cranfield.ac.uk
 *
 * Developed using NIST STEP Class Library
 *
 * Available for download from
 *
 * http://www.mel.nist.gov/msidstaff/sauder/SCL.htm#download
 *
 * Some sections adapted from SCL basic examples
 *
 * In particular treg.cc, Written by Ian Soboroff, NIST. June, 1994
 *
 */

/* A switch for tests.h, because we don't need to schema header file */
#define DONT_NEED_HEADER
#include "tests.h"
#include "midsurf.h"
#include "time.h"
#include <instmgr.h>
#include <string>

MidSurface mymid;

// This function takes a STEPentity as input, and cycles through the
// attributes of the entity collecting topology information
// The function captures relationship information for the following
// geometry entities
//              FACE_SURFACE
//              FACE_OUTER_BOUND
//              FACE_BOUND
//              EDGE_LOOP
//              ORIENTED_EDGE
//              EDGE_CURVE
//              ADVANCED_FACE
//              SHELL_BASED_SURFACE_MODEL
//              B_SPLINE_SURFACE_WITH_KNOTS
//              B_SPLINE_CURVE_WITH_KNOTS
//              BOUNDED_SURFACE
//              CARTESIAN_POINT
//              PLANE
//              CYLINDRICAL_SURFACE
//              AXIS2
//              DIRECTION
//

int ConvertLabelToInteger(const char *Str2)
        {
                char c2 = ' ';
                char newtmp[16];
                int Ncount;
                int stringlength2 = strlen(Str2);
                for (Ncount = 0; Ncount < stringlength2; Ncount++)
                {
                        newtmp[Ncount] = Str2[Ncount+1];
                }
                int Index2 = atoi(newtmp);
                return Index2;
        }


MidSurface OutputEntity(STEPentity *entdesc, InstMgr instance_list, MidSurface
mymidsub, int output, ofstream& outputmap )
{
    int attrCount = entdesc->AttributeCount();
    const char *MyEntityName = entdesc->EntityName();

        ostrstream myenttype;
```

```
        myenttype << entdesc->EntityName();

//set up strings of all known types for type checking
//
        const string my_edge_curve = "Edge_Curve";
        const string my_oriented_edge = "Oriented_Edge";
        const string my_edge_loop = "Edge_Loop";
        const string my_face_bound = "Face_Bound";
        const string my_face_outer_bound = "Face_Outer_Bound";
        const string my_face_surface = "Face_Surface";
        const string my_advanced_face = "Advanced_Face";
        const string my_shell_based_surface_model = "Shell_Based_Surface_Model";
        const string my_b_spline_surface_with_knots = "B_Spline_Surface_With_Knots";
        const string my_b_spline_curve_with_knots = "B_Spline_Curve_With_Knots";
        const string my_bounded_surface = "Bounded_Surface";
        const string my_cartesian_point = "Cartesian_Point";
        const string my_plane = "Plane";
        const string my_cylindrical_surface = "Cylindrical_Surface";
        const string my_axis2 = "Axis2_Placement_3d";
        const string my_direction = "Direction";


        int needOutput = 0;  // true if we need to output the value
                             // that is, if it's anything but 'none'

        int checkAggr= 0; // true if we want to check the values of aggregates for this
ent type
                         // false if we want to skip the aggregatesfor this type
        int write_mapping = 0; //flag to tell program to write entity to the mapping file
        int write_face_geom = 0; // flag to tell program to set planar and cylindrical
faces
        ostrstream valstr;

        valstr << entdesc->EntityName();
        valstr << ends; // clean terminate valstr

//
// Output attributes for all entities that have a geometry type of interest
//
        if (valstr.str() == my_edge_curve)
        {
                int edgeno = entdesc-> StepFileId();
                needOutput = 1;
                checkAggr = 1;
                if (needOutput == output)
                        mymidsub.add_edge(edgeno);
        }
        else if (valstr.str() == my_oriented_edge)
        {
                needOutput = 0;
                checkAggr = 1;
                }
        else if (valstr.str() == my_edge_loop)
        {
                needOutput = 1;
                checkAggr = 1;
                if (needOutput == output)
                        mymidsub.add_edge_loop(entdesc->StepFileId());
                }
        else if (valstr.str() == my_face_bound)
        {
                needOutput = 1;
                checkAggr = 1;
                if (needOutput == output)
                        mymidsub.add_bound(entdesc->StepFileId());

                }
        else if (valstr.str() == my_face_outer_bound)
        {
                needOutput = 1;
                checkAggr = 1;
                if (needOutput == output)
                        mymidsub.add_bound(entdesc->StepFileId());
                }
        else if (valstr.str() == my_face_surface)
        {
                needOutput = 1;
                checkAggr= 1;
                if (needOutput == output)
                        mymidsub.add_face(entdesc-> StepFileId());
                }
         else if (valstr.str() == my_advanced_face)
        {
                needOutput = 1;
                checkAggr = 1;
                if (needOutput == output)
                        mymidsub.add_face(entdesc-> StepFileId());
                write_mapping = 1;
                write_face_geom = 1;
                }
        else if (valstr.str() == my_shell_based_surface_model)
        {
                needOutput = 1;
                checkAggr = 1;
                if (needOutput == output)
                        mymidsub.add_shell(entdesc-> StepFileId());
        }
        else if (valstr.str() == my_b_spline_surface_with_knots)
        {
                needOutput = 1;
                checkAggr = 3;
         }
       else if (valstr.str() == my_b_spline_curve_with_knots)
        {
                needOutput = 1;
                checkAggr = 0;
        }
       else if (valstr.str() == my_bounded_surface)
        {
                needOutput = 1;
                checkAggr = 0;
        }
        else if (valstr.str() == my_direction)
        {
                needOutput = 1;
                checkAggr = 0;
        }
      else if (valstr.str() == my_cartesian_point)
        {
                needOutput = 1;
                checkAggr = 2;
        }
        else
//
// Otherwise do not output anything
//
                needOutput = 0;


//
//    Start at the first attribute
//

    entdesc->ResetAttributes();

    const TypeDescriptor *desc = entdesc->eDesc;
    SCLstring u, v;
           const string u_deg = "u_degree";
        const string v_deg = "v_degree";
        const string my_control_points_list = "control_points_list";
        const string my_face_geometry = "face_geometry";
        const string my_position = "position";
        const string my_axis = "axis";
        const string my_direction_ratios = "direction_ratios";
//        const string my_plane = "Plane";
//        const string my_cylindrical_surface = "Cylindrical_Surface";
    STEPattribute *attr = entdesc->NextAttribute();

//
// Cycle through all the attributes in the current STEPentity to get faces and
//edges
```

```
//
    if (needOutput == output)
    {

    while (attr != 0)
    {
        const AttrDescriptor *attrDesc = attr->aDesc;
        const char *EdgeEl = ("edge_element");
        const char *Bound = ("bound");
        const char *AdvFace = ("advanced_face");

        //
        // For attributes that are of type aggregate need to get all the
        // members of the aggregate
        //

        SCLstring tmp;
        if (attrDesc -> IsAggrType() && checkAggr == 2)
 // For Cartesian points output the X Y an Z coordinates// For Cartesian points output
the X Y an Z coordinates
        {
        real xval, yval, zval;

           STEPaggregate *XYZAggr = attr->ptr.a;
           RealNode* RNode = (RealNode*)XYZAggr->GetHead();
             xval=((RealNode*)RNode)->value;
            RNode = (RealNode*)RNode  -> NextNode();
            yval = ((RealNode*)RNode)->value;
            RNode = (RealNode*)RNode -> NextNode();
            zval = ((RealNode*)RNode)->value;
            mymidsub.add_vertex(entdesc -> StepFileId(), xval, yval, zval);
        }
        if (write_face_geom == 1 && attrDesc -> Name()== my_face_geometry)
        {
        SCLstring s2;

                int pstepid = ConvertLabelToInteger(attr->asStr(s2));
                MgrNode* MgrTmp2 = instance_list.FindFileId(pstepid);
                STEPentity  *PointedEntity = instance_list.GetSTEPentity(MgrTmp2);

           if (PointedEntity->EntityName()== my_plane)
                {
                        mymidsub.set_planar_face(entdesc -> StepFileId());
                        //Code Segment to set face normal dir. for planar faces
                        PointedEntity->ResetAttributes();
                        STEPattribute *SkipPosition = PointedEntity->NextAttribute();
                        STEPattribute *Position = PointedEntity->NextAttribute();
                        int stepid1 = ConvertLabelToInteger(Position->asStr(s2));
                        MgrNode* MgrTmp11 = instance_list.FindFileId(stepid1);
                        STEPentity  *PointedEntity1 =
instance_list.GetSTEPentity(MgrTmp11);
                        PointedEntity1->ResetAttributes();
                        STEPattribute *SkipAxis1 = PointedEntity1->NextAttribute();
                        STEPattribute *SkipAxis2 = PointedEntity1->NextAttribute();
                        STEPattribute *Axis = PointedEntity1->NextAttribute();
                        int stepid2 = ConvertLabelToInteger(Axis->asStr(s2));
                        MgrNode* MgrTmp12 = instance_list.FindFileId(stepid2);
                        STEPentity  *PointedEntity2 =
instance_list.GetSTEPentity(MgrTmp12);
                        PointedEntity2->ResetAttributes();
                        STEPattribute *SkipDirection1 = PointedEntity2->NextAttribute();
                        STEPattribute *Direction = PointedEntity2->NextAttribute();
                        float xdir, ydir, zdir;
                        STEPaggregate *NormalDirAggr = Direction->ptr.a;
                        RealNode* RNode = (RealNode*)NormalDirAggr->GetHead();
                        xdir=((RealNode*)RNode)->value;
                        RNode = (RealNode*)RNode  -> NextNode();
                        ydir = ((RealNode*)RNode)->value;
                        RNode = (RealNode*)RNode -> NextNode();
                        zdir = ((RealNode*)RNode)->value;
                        mymidsub.set_face_normal(entdesc->StepFileId(),xdir, ydir,zdir);
                }
           if (PointedEntity->EntityName()== my_cylindrical_surface)
                {
                        mymidsub.set_cylindrical_face(entdesc -> StepFileId());
                }
                write_face_geom = 0;
        }

        if (attrDesc -> IsAggrType() && checkAggr == 3)
        {

         if (atoi(u)== 1 && atoi(v) == 1 && attrDesc -> Name() == my_control_points_list)
          {
//Following code taken fromo Geant4, to overcome bug in NIST toolkit


                STEPaggregate *Aggr = attr->ptr.a;
                GenericAggregate* gAggr  =  (GenericAggregate*)attr->ptr.a;
                // Get control points

                      int cols = 2;
                      int rows = 2;

                      char tmp[16];
                      SCLstring s;
                      const char *Str = Aggr->asStr(s);

                      int stringlength = strlen(Str);
                     int ControlPoints[rows][cols];
                      RealAggregate rationalAggr;
                      int Count = 0;
                      for(int a=0;a<rows;a++)
                      for(int b=0;b<cols;b++)
                      {
                              // get points

                             // temp version until the NIST toolkit can handle two
dimensional aggregates
                             // The string Str contains the STEP file id:s of the
underlying point
                             // entities so well have to parse the string to get them
out...arghhh!
                               char c = ' ';
                                  // Loop to find the entities


                                 // Fill points
                                 //Temporary solution until the STEP toolkit has been
updated:
            while(c != '#')
              {
                c = Str[Count];
                Count++;
                if(Count>stringlength)
                  {
                    cout << "\nString index overflow in G4ControlPoints:116";
                    exit(0);
                  }
              }

            c = Str[Count];
            int Index=0;

            while(c != ',' && c != ')')
              {
                tmp[Index]=c;
                Index++;
                Count++;
                c = Str[Count];
              }
//              cout << " value of c is " << c << endl;
            tmp[Index]='\0';
            Index = atoi(tmp);
            cout << "Value of Index is " << Index << endl;
            MgrNode* MgrTmp = instance_list.FindFileId(Index);
            ControlPoints[a][b]= Index; //is this right should it be entity?

          mymidsub.add_vertex_to_face(entdesc -> StepFileId(), Index);
          }
        }
        }
```

```
        if (attrDesc-> IsAggrType() && checkAggr == 1)
        {
                SCLstring s;

                //Following code section adapted from Geant4 software from CERN
                // (http://cern.ch.geant4/
                //Filename: G4BSplineCurveWithKnotsCreator.cc
                //
                // Get edge names from edge_loop
                // Loop to find the entities
                // Fill array of entities
                // Temporary solution until the STEP toolkit has been updated:
                char c = ' ';
                STEPaggregate *Aggr = attr->ptr.a;
                char tmp[16];
                int list_of_aggregates[50];
                const char *Str = attr->asStr(s);
                int Count=0;
                int nbpoint = 0;

                int stringlength = strlen(Str);

                while(c != ')')
                {
                        while(c != '#')
                                {
                                        c = Str[Count];

                                        Count++;

                                        if(Count>stringlength)
                                        {
                                        cout << "\nString index overflow in aggregate
array";
                                        exit(0);
                                }
                }

                c = Str[Count];
                int Index=0;


                while(c != ',' && c != ')')
                {
                        tmp[Index]=c;
                        Index++;
                        Count++;
                        c = Str[Count];
                }

                tmp[Index]='\0';

                Index = atoi(tmp);
                list_of_aggregates[nbpoint] = Index;
                MgrNode* MgrTmp = instance_list.FindFileId(Index);

                STEPentity  *Entity = instance_list.GetSTEPentity(MgrTmp);
                Entity -> ResetAttributes();


                STEPattribute *tmpAttr = Entity->NextAttribute();
                int num;
                while (tmpAttr != 0)
                {
                        int stepid;
                        const AttrDescriptor *ad = tmpAttr->aDesc;
                        SCLstring tmp2;

                        if ((strcmp (ad -> Name(), EdgeEl)) == 0)
                        {
                                stepid = ConvertLabelToInteger(tmpAttr->asStr(tmp2));
                                mymidsub.add_edge_to_loop(entdesc->StepFileId(), stepid);

                        }
                        else if ((strcmp (ad -> Name(), Bound)) == 0)
                        {
                                stepid = ConvertLabelToInteger(tmpAttr->asStr(tmp2));
                                mymidsub.add_bound_to_face(entdesc->StepFileId(), Index);
                                mymidsub.add_edge_loop_to_bound(Index, stepid);
                        }

                        tmpAttr = Entity -> NextAttribute();

                }

                nbpoint++;
                }
        }
        else if ((strcmp (attrDesc->Name(), Bound)) == 0)
        {
        }

        else

        valstr << ends;  // flush and null-terminate the stream
        char *val = valstr.str();  // fix stream into char* string
        SCLstring tmp3;
        if (attrDesc -> Name()== u_deg) u = attr ->asStr(tmp3);
        if (attrDesc -> Name()== v_deg) v = attr->asStr(tmp3);
        const string face_geom = "face_geometry";
        if (attrDesc -> Name()== face_geom)
        {
                int  face_geom = ConvertLabelToInteger(attr->asStr(tmp3));
                mymidsub.add_face_geometry_to_face(entdesc -> StepFileId(), face_geom);
        }
        const string name = "name";
        if (attrDesc -> Name() == name && write_mapping == 1)
                        {
               mymidsub.add_face_ideas_id(attr -> asStr(tmp), entdesc -> StepFileId());

               outputmap << attr -> asStr(tmp) << endl;
                write_mapping = 0;
                        }
        attr = entdesc->NextAttribute();
    }

    }

   return mymidsub;
}


int main(int argc, char *argv[])
{
    int using_outfile = 0;

// Open file to write mapping table to
char* mapping = "mapping-file.txt";
char* timingfile = "timing.txt";
ofstream outputmap(mapping, ios::out);
ofstream timing(timingfile, ios::out);

//Set up timing routines
        time_t tim0=time(NULL);
        clock_t cloc0 =  clock();
        char *timeStr0 = ctime(&tim0);
  timing << " Program Start time: " << timeStr0 << endl;
        timing << " Clock Start " <<  cloc0 << endl;
    if (argc > 2)
    {
        cout << "Syntax:   geomfromstep [filename]" << endl;
        exit(1);
    }
    else if (argc > 1)
        using_outfile = 1;  // output filename is in argc[1]

    // This has to be done before anything else.  This initializes
    // all of the registry information for the schema you are using.
    // The SchemaInit() function is generated by fedex_plus... see
    // extern statement above.

    Registry *registry = new Registry(SchemaInit);
```

```
    // The nifty thing about the Registry is that it basically keeps a list
    // of everything in your schema.  What this means is that we can go
    // through the Registry and instantiate, say, one of everything, without
    // knowing at coding-time what entities there are to instantiate.  So,
    // this test could be linked with other class libraries produced from
    // other schema, rather than the example, and run happily.

    InstMgr instance_list;
    STEPfile *sfile = new STEPfile(*registry, instance_list);

    // The STEPfile is actually an object that manages the relationship
    // between what's instantiated in the instance manager, and how that
    // information gets passed to the outside, e.g., a file on disk.
    //
    // Read an instance file
    //
    cout << "\n### Reading from input file " << argv[1] << endl;
    sfile->ReadExchangeFile(argv[1]);

    // Check how many instances are found in the file
    //
    int num_ents = instance_list.InstanceCount();
    cout << "\n### Number of entities is " << num_ents << endl;
//    int num_ents = registry->GetEntityCnt();

    STEPentity** SEarray = new STEPentity*[num_ents];

    // "Reset" the Schema and Entity hash tables... this sets things up
    // so we can walk through the table using registry->NextEntity()

    registry->ResetSchemas();
    registry->ResetEntities();

    // Print out what schema we're running through.

    const SchemaDescriptor *schema = registry->NextSchema();
    cout << "Collecting Instance entities for schema " << schema->Name() << endl;

    // "Loop" through the instance file , processing all entities.

    //    const ApplicationInstance *ent = instance;   // needs to be declared const...
    const EntityDescriptor *entdesc ;

    for (int i=0; i < num_ents; i++)
    {

        const STEPentity *ent = instance_list.GetSTEPentity(i);
        SCLstring tmp1;

//      Add instance to instance list
        SEarray[i] = instance_list.GetSTEPentity(i);


// Output attribute information for current instance
        int output = 1;
        mymid = OutputEntity(SEarray[i], instance_list, mymid, output, outputmap);
    }

//Close the mapping file
outputmap.close();
//Set up timing routines
        time_t tim1=time(NULL);
        clock_t cloc1 =  clock();
        char *timeStr = ctime(&tim1);
        timing << " Time 1 (after creating the data structure) " << timeStr << endl;
        timing << " Clock 1  " <<  cloc1-cloc0 << endl;
    mymid.update_adj_matrix();
         time_t tim2=time(NULL);
         clock_t cloc2 =  clock();
        timeStr = ctime(&tim2);
        timing << " Time 1 (after populating the adjacency matrix) " << timeStr << endl;
        timing << " Clock 1  " <<  cloc2-cloc1  << endl;
    cout << "" << endl;
    cout << "SUMMARY of recognised Features" << endl;
    cout << "------------------------------" << endl << endl;
```

```
        time_t tim3=time(NULL);
        clock_t cloc3 =  clock();
        timeStr = ctime(&tim3);
        timing << " Time 3 (after printing the adjacency matrix) " << timeStr << endl;
        timing << " Clock 3  " <<  cloc3-cloc2 << endl;
    mymid.create_feat_file();
    mymid.find_holes();
        time_t tim4=time(NULL);
        clock_t cloc4 =  clock();
        timeStr = ctime(&tim4);
        timing << " Time 4 (after finding the face features) " << timeStr << endl;
        timing << " Clock 4  " <<  cloc4-cloc3  << endl;
    mymid.find_stiffeners();
        time_t tim5=time(NULL);
        clock_t cloc5 =  clock();
        timeStr = ctime(&tim5);
        timing << " Time 5 (after finding the stiffeners) " << timeStr << endl;
        timing << " Clock 5  " <<  cloc5-cloc4  << endl;
    mymid.find_2nd_order_junctions();
         time_t tim6=time(NULL);
        clock_t cloc6 =  clock();
        timeStr = ctime(&tim6);
        int timeStrtot = tim6-tim0;
        timing << " Time 5 (after finding the face angles) " << timeStr << endl;
        timing << " Clock 5  " <<  cloc6-cloc5  << endl;
        timing << " Total Time is " << timeStrtot << endl;
        timing << " Total Clock is " <<  cloc6-cloc0  << endl;
    mymid.output_high_order_edges();
    mymid.generate_step_overlay();
    timing.close();

}
```

```
;;  Knowledge Based Manufacturing Advisor
;;
;;  Helen Lockett
;;  December 2004
;;
;;
;; MODIFICATION LOG
;;
;; 8th December 2004
;; Changed structure to incorporate all part design parameters into "part design"
;; Modified all rules to test agains content of part design for current values
;;
;; 21st December 2004
;; Began to update rules to use the reference IDs from my rule base.
;; Need to tidy up the max min wall thickness rule - it is too simplistic and gives
wrong answer!
;; Definition of knowledge templates
;;
;; 6 - 7 Jan 2005
;; Continued tidying up code to use new structure
;; Continued updating rules to match those in the database
;;
;; Would be nice to print a report telling the user what inputs they gave before giving
advice
;;
;; note that the "design level" setting is now notreally used.  Would  be good to remove
completely
;;

;; 16th January (v12)
;; Many changes 14th Jan - good improvements to overall structure, and all generic rules
working
;;
;;
;; Further changes on 16 Jan have caused sequencing problems - removed the last of the
design level
;; Added code to chekc that all features are processed before proceding with the advice
;; Now the "collate requirements" repeatedly reruns - I cannot see why
;; but I note that the part-design structure isnow very long
;; maybe better to separate the part design from the processing - have a new
;; structure with all the processing instructions?
;;
;; 18th January (v13)
;; Processing errors mostly fixed
;; Implemented the control structure to manage progress
;; Need to write results to a file
;; and to revisit the processing sequence for feature models.
;;
;; 19th January (v14)
;; Results now write to an external file
;; Some improvements to formatting
;; added code to automatically close the external file - mostly works
;; tried to improv control structure to set minimum wall thickness before processing
features
;; but caused a problem - code doesn't run!
;;
;; 21st January (v14)
;; fixed problems above, code now seems to run reliable again
;;
;; v15
;; Working  on the feature specific rules to match to rule base.
;;
;; v16
;; Improved the feature file  to support high order junctions and main wall faces
;;
;;

;;


;;DECLARATIONS

(deftemplate part-design
             (slot part-name (type STRING))
             (slot process (type SYMBOL) (allowed-symbols unset unknown injection-
moulding sand-casting die-casting casting))
```

```
             (slot material-type (type SYMBOL) (allowed-symbols unset metal plastic
unknown))
             (slot material-name (type SYMBOL) (allowed-symbols unset acetal acrylic
long_fibre_reinforced aluminium magnesium steel zinc unknown) (default unset))
             (slot feature-model (type SYMBOL) (allowed-symbols yes no  unset))
             (slot max-wall-tk (type FLOAT) (default 0.0))
             (slot min-wall-tk (type FLOAT) (default 0.0))
             (slot max-main-wall-tk (type FLOAT) (default 0.0))
             (slot min-main-wall-tk (type FLOAT) (default 0.0))
             (slot draft-angle (type FLOAT) (default 0.0))
             (slot radius (type FLOAT) (default 0.0))
;;             (slot design-req (type SYMBOL) (allowed-symbols yes no unset) (default
unset))
;;             (slot thickness-set (type SYMBOL) (allowed-symbols yes unset) (default
unset))
             (slot strength (type FLOAT) (default 0.0))
             (slot appearance (type FLOAT) (default 0.0))
             (slot too-thick (type SYMBOL) (allowed-symbols yes no) (default no))
;;             (slot all-features-proc (type SYMBOL) (allowed-symbols yes no) (default
no))
              )

(deftemplate control
            (slot design-req (type SYMBOL) (allowed-symbols yes no unset) (default
unset))
            (slot thickness-set (type SYMBOL) (allowed-symbols yes unset) (default
unset))
            (slot mw-thickness-set (type SYMBOL) (allowed-symbols yes unset) (default
unset))
            (slot all-features-proc (type SYMBOL) (allowed-symbols yes no) (default no))
            (slot set-default-wall (type SYMBOL) (allowed-symbols yes no) (default no))
            (slot generic-finished (type SYMBOL) (allowed-symbols yes no) (default no))
)

;;(deftemplate design-req
;;             (slot property (type SYMBOL))
;;             (slot cert (type FLOAT)))

(deftemplate design-params
             (slot initialise(type SYMBOL) (allowed-symbols YES NO) (default NO))
             (slot max-wall-tk (type FLOAT) (default 0.0))
             (slot min-wall-tk (type FLOAT) (default 0.0))
             (slot draft-angle (type FLOAT) (default 0.0))
             (slot radius (type FLOAT) (default 0.0))
;;             (slot variable-wall-tk (type SYMBOL))
)

(deftemplate process
             (slot proc-name (type SYMBOL))
             (slot max-wall-tk (type FLOAT))
             (slot min-wall-tk (type FLOAT))
             (slot min-draft-angle (type FLOAT))
             (slot typ-draft-angle (type FLOAT))
             (slot fillet-rad (type FLOAT))
             (slot rec-rib-prop (type FLOAT))
             (slot rec-buttress-prop (type FLOAT))
             (slot section-change-ratio (type FLOAT)))

(deftemplate material
            (slot material-name (type SYMBOL) (allowed-symbols aluminium magnesium steel
zinc acrylic acetal long_fibre_reinforced))
            (slot process-name (type SYMBOL))
            (slot max-wall-tk (type FLOAT))  ;mm
            (slot min-wall-tk (type FLOAT))  ;mm
            (slot min-draft-angle (type FLOAT)) ;degrees
            (slot shrinkage (type FLOAT))       ;percent
            (slot yield-strength (type FLOAT))  ;MPa
            (slot thermal-conductivity(type FLOAT))
            (slot density (type FLOAT)))  ;g/cm3

(deftemplate design-guideline
             (slot advice-string (type STRING))
             (multislot material (type SYMBOL))
             (multislot process (type SYMBOL))
             (slot certainty (type FLOAT)))
```

```
(deftemplate feature-model
             (slot available (type SYMBOL)(allowed-symbols YES NO))
             (slot filename (type STRING))
;; Should remove next line
             (slot thickness (type FLOAT))
             (slot set-default (type SYMBOL)(allowed-symbols YES NO) (default NO)))

(deftemplate feature
             (slot feat-number (type INTEGER))
             (slot id (type STRING))
             (slot feat-type (type SYMBOL) (allowed-symbols rib boss fin buttress hole
ho-edge main-wall))
             (slot thickness (type FLOAT))
             (slot checked-tk (type SYMBOL) (allowed-symbols YES NO) (default NO))
             (slot checked-mainwall (type SYMBOL) (allowed-symbols YES NO) (default NO))
             (slot checked-wall (type SYMBOL) (allowed-symbols YES NO) (default NO))
             (slot has-thickness (type SYMBOL) (allowed-symbols YES NO) (default NO))
             (slot edge-order (type INTEGER)))

;;Define the facts about manufacturing processes and materials

;;FACTS

;; NEED TO CHECK VALUES FOR EACH PROCESS
;;
(deffacts processes
;; SCF1, SCF2, SCF3,
          (process (proc-name sand-casting) (max-wall-tk 127.0) (min-wall-tk 6.35)
(section-change-ratio 4.0) (min-draft-angle 1.0)
                   (typ-draft-angle 2.0) (rec-rib-prop 1.0) (rec-buttress-prop 1.0))
           (process (proc-name die-casting) (max-wall-tk 5.08) (min-wall-tk 0.5)(section-
change-ratio 4.0) (min-draft-angle 0.25)
                   (typ-draft-angle 1.0) (rec-rib-prop 1.0)(rec-buttress-prop 1.0))
;;section change ratio not available,used sand value
           (process (proc-name injection-moulding) (max-wall-tk 5.0) (min-wall-tk 2.0)
(section-change-ratio 3.0)(min-draft-angle 0.5)
                   (typ-draft-angle 1.0) (rec-rib-prop 0.6)(rec-buttress-prop 0.5))
           (process (proc-name casting) (max-wall-tk 127.0) (min-wall-tk 0.5) (min-draft-
angle 0.25)
                    (typ-draft-angle 1.0) (rec-rib-prop 0.8)(rec-buttress-prop 1.0)))

(deffacts materials
          (material (material-name aluminium) (process-name sand-casting) (min-wall-tk
2.54) (max-wall-tk 127.0)) ;;max generic for process
          (material (material-name aluminium) (process-name die-casting) (min-wall-tk
0.9) (max-wall-tk 6.35))
          (material (material-name magnesium) (process-name sand-casting) (min-wall-tk
3.3)(max-wall-tk 127.0)) ;;max generic for process
          (material (material-name steel) (process-name sand-casting) (min-wall-tk
3.3)(max-wall-tk 127.0)) ;;max generic for process
          (material (material-name zinc) (process-name die-casting) (min-wall-tk
0.6)(max-wall-tk 9.65))
          (material (material-name long_fibre_reinforced) (process-name injection-
moulding) (min-wall-tk 1.9)(max-wall-tk 25.4))
          (material (material-name acetal) (process-name injection-moulding) (min-wall-
tk 0.8)(max-wall-tk 3.6))
          (material (material-name acrylic) (process-name injection-moulding) (min-wall-
tk 0.6) (max-wall-tk 3.0)))


;;RULES
;;
;;Initialise program.
;;


(defrule initialise-program  ""
         (not (part-design (process ?)))
         (not (part-design (material-type ?)))

         =>
         (printout
                  t crlf
                  "******************** WELCOME TO THE KNOWLEDGE BASED MOULDING
ADVISOR********************"
                  t crlf
```

```
                  "******************************WRITTEN BY HELEN
LOCKETT***********************************"
                  t crlf

                  "
**************************************************************                "
                  t crlf
                  )
         (printout
              t crlf
                  "QUESTION: Enter a name for this session"
                  t crlf)
            (bind ?name (read))
          (assert (part-design (part-name ?name)(process unset)(material-type
unset)(material-name unset)(feature-model unset)))
          (assert (design-params (initialise YES) (max-wall-tk 0.0) (min-wall-tk 0.0)))
            (assert (control (design-req unset)))
          (assert (design-level 0))
            (open "Z:\results.txt" writefile "w")
    (printout
        t crlf
                "          INFORMATION: The manufacturing advice will be written to the
results file results.txt"
        t crlf)
    (printout writefile
                  t crlf
                  "******************** OUTPUT FROM KNOWLEDGE BASED MOULDING
ADVISOR********************"
                  t crlf

                  "******************************WRITTEN BY HELEN
LOCKETT***********************************"
                  t crlf

                  "       *******************MANUFACUTRING ADVICE FOR PART " ?name
"**********************                "
                  t crlf

                  )
                    )

;;
;; Rule to elicit the material type
;;

(defrule select-material-type   ""
         (part-design (part-name ?pname) (process unset) (material-type unset))
         ?mypart <- (part-design (part-name ?pname)(material-type ?mat))
         =>
         (printout
                 t crlf
                  "QUESTION: Is the part to be manufactured from PLASTIC or METAL or
UNKNOWN?"
                  t crlf)
         (modify ?mypart (material-type (read))))

;;
;; Rule to elicit the specific material (metallic)
;;

(defrule select-material-type-metallic  ""
         (part-design (part-name ?pname)(material-name unset)(material-type metal))
         ?mypart <- (part-design (part-name ?pname)(material-type ?mat))
         =>
         (printout
                 t crlf
                  "QUESTION: Is the part to be manufactured from aluminimum, magnesium,
steel or zinc?"
                  t crlf)
         (bind ?newmat (read))
         (modify ?mypart (material-name ?newmat))
         (printout
                t crlf
                "          INFORMATION: The material has been set to " ?newmat
                t crlf)
        (printout  writefile
                t crlf
                "          INFORMATION: The material has been set to " ?newmat
                t crlf)
)

 ;;
;; Rule to elicit the specific material (plastic)
;;

(defrule select-material-type-plastic  ""
         (part-design (part-name ?pname)(material-name unset)(material-type plastic))
         ?mypart <- (part-design (part-name ?pname)(material-type ?mat))
         =>
         (printout
                 t crlf
                  "QUESTION: Is the part to be manufactured from acetal, acrylic, or
long_fibre_reinforced or unknown?"
                  t crlf)
         (bind ?newmat (read))
         (modify ?mypart (material-name ?newmat))
         (printout
                t crlf
                "          INFORMATION: The material has been set to " ?newmat
                t crlf)
        (printout writefile
                t crlf
                "          INFORMATION: The material has been set to " ?newmat
                t crlf)

         )
;;
;; Rule to elicit process if material is not specified
;;

(defrule select-process-type   ""
         (part-design (part-name ?pname) (material-type unknown) (process unset))
         ?mypart <- (part-design (part-name ?pname)(material-type ?mat))
         =>
         (printout
                  t crlf
                  "QUESTION: Is the manufacturing process INJECTION-MOULDING, SAND-
CASTING, DIE-CASTING or UNKNOWN?"
                  t crlf)
         (bind ?newproc (read))

          (if (eq ?newproc injection-moulding)
                then
                (modify ?mypart(process ?newproc) (material-type plastic))
                  (printout
                  t crlf
                  "          INFORMATION: The process has been set to " ?newproc
                  t crlf
                  "          ERRORCHECK: The material-type has been set to plastic"
                  t crlf)
                 (printout writefile
                  t crlf
                  "          INFORMATION: The process has been set to " ?newproc
                  t crlf )
                 else
                   (modify ?mypart(process ?newproc) (material-type metal))
                  (printout
                  t crlf
                  "          INFORMATION: The process has been set to " ?newproc
                  t crlf
                  "          ERRORCHECK: The material-type has been set to metal"
                  t crlf)
                 (printout writefile
                  t crlf
                  "          INFORMATION: The process has been set to " ?newproc
                  t crlf))
)

;;
;;Rule to check whether a feature model is available and to read data from the feature
file
```

```
;;

(defrule read-feature-model ""
       (part-design (part-name ?pname) (feature-model unset))
       (not (part-design (process unset)))
       (not (part-design (material-type unset)))
       (not (feature (feat-number ?)))
       ?mypart <- (part-design (part-name ?pname)(feature-model ?val))

         =>
            (printout
                  t crlf
                  "QUESTION: Is there a feature model available for the design? yes/no"
                  t crlf)
             (bind ?avail (read))
            (if (eq ?avail yes)
                then
                (printout
                  t crlf
                  "QUESTION: Enter the name of the file containing the feature model"
                  t crlf
                  "(for testing use feat-jun05.txt)"
                  t crlf)
             (modify ?mypart (feature-model yes))
;; Read the feature information from a file
           (bind ?file (read))
           (assert (feature-model (available YES)(filename ?file)))
         (load-facts ?file)
           (printout writefile
                  t crlf
                  "          INFORMATION: A feature model has been read from the file "
?file
                  t crlf)

            else (assert (feature-model (available NO)))
                 (modify ?mypart (feature-model no))
          ))



;;
;; Rule to define baseline wall thickness for part with a feature model
;;

(defrule define-baseline-wall-thickness ""
        (part-design (part-name ?pname)(feature-model yes) (max-wall-tk ?maxw) (min-wall-
tk ?minw)(max-main-wall-tk ?maxw) (min-main-wall-tk ?minw))
        (control (set-default-wall no))
        ?mypart <- (part-design (part-name ?pname)(feature-model yes) (max-wall-tk
?maxw) (min-wall-tk ?minw)(max-main-wall-tk ?maxmw) (min-main-wall-tk ?minmw))
        ?model1 <- (feature-model (thickness ?wall-tk) (available YES) (set-default NO) )
        ?mycon <- (control (set-default-wall ?sdw))
        =>
        (modify ?mycon (set-default-wall yes))
        (modify ?mypart (max-wall-tk ?wall-tk)(min-wall-tk ?wall-tk)(max-main-wall-tk
?wall-tk)(min-main-wall-tk ?wall-tk))
        (printout
                  t crlf
                  "          INFORMATION: The nominal wall thickness has been set to "
?wall-tk  " (based on value from feature file)"
                  t crlf)
        (printout writefile
                  t crlf
                  "          INFORMATION: The nominal wall thickness has been set to "
?wall-tk  " (based on value from feature file)"
                  t crlf)

)


;;
;; Rule to update the part main wall thickness  from the feature data
;;

(defrule define-main-wall-tk ""
                  (control (set-default-wall yes))
         ?mypart <- (part-design (part-name ?pname) (process ?proc1) (feature-model yes)
(max-main-wall-tk ?maxw) (min-main-wall-tk ?minw))
         ?model1 <- (feature-model (thickness ?wall-tk) (available YES))
         ?feat1 <- (feature (feat-type main-wall) (feat-number ?num) (thickness ?feat-
tk) (checked-mainwall NO)(has-thickness YES))
             ?mycon <- (control (mw-thickness-set ?))
         =>
            (printout writefile t crlf "          INFORMATION: Feature " ?num " is of
type main wall and has thickness " ?feat-tk "  mm" t crlf)
         (printout t crlf "          ERRORCHECK: In define-design-params-from-file
routine " t crlf
                          "          ERRORCHECK: Current value of max-main-wall-tk is "
?maxw " and min-main-wall-tk is " ?minw t crlf
                          "          ERRORCHECK: Value of ?feat1 is " ?feat-tk t crlf)
                (modify ?feat1 (feat-number ?num) (checked-mainwall YES))
         (if (> ?feat-tk ?maxw)
                then
                        (modify ?mypart (max-main-wall-tk ?feat-tk)))

           (if (< ?feat-tk ?minw)
                then
                        (modify ?mypart (min-main-wall-tk ?feat-tk)))
    (modify ?mycon  (mw-thickness-set yes))
         )

;;
;; Rule to update the part design parameters from the feature data
;;

(defrule define-design-params-from-file ""
                  (control (set-default-wall yes))
         ?mypart <- (part-design (part-name ?pname) (process ?proc1) (feature-model yes)
(max-wall-tk ?maxw) (min-wall-tk ?minw))
         ?model1 <- (feature-model (thickness ?wall-tk) (available YES))
         ?feat1 <- (feature (feat-type ?feat) (feat-number ?num) (thickness ?feat-tk)
(checked-tk NO) (checked-wall NO)(has-thickness YES))
             ?mycon <- (control (thickness-set ?))
         =>
            (printout writefile t crlf "          INFORMATION: Feature " ?num " is of
type " ?feat " and has thickness " ?feat-tk "  mm" t crlf)
         (printout t crlf "          ERRORCHECK: In define-design-params-from-file
routine " t crlf
                          "          ERRORCHECK: Current value of max-wall-tk is " ?maxw
" and min-wall-tk is " ?minw t crlf
                          "          ERRORCHECK: Value of ?feat1 is " ?feat-tk t crlf)
                (modify ?feat1 (feat-number ?num) (checked-wall YES))
         (if (> ?feat-tk ?maxw)
                then
                        (modify ?mypart (max-wall-tk ?feat-tk)))

           (if (< ?feat-tk ?minw)
                then
                        (modify ?mypart (min-wall-tk ?feat-tk)))
    (modify ?mycon  (thickness-set yes))
         )
;;
;; Rule to ensure that features with no thickness are skipped to allow progress to the
advice phase
;;

(defrule skip-no-tk-features ""
                  (control (set-default-wall yes))
         ?mypart <- (part-design (part-name ?pname) (process ?proc1) (feature-model yes)
(max-wall-tk ?maxw) (min-wall-tk ?minw))
           ?model1 <- (feature-model (thickness ?wall-tk) (available YES))
         ?feat1 <- (feature (feat-type ?feat) (feat-number ?num) (thickness ?feat-tk)
(checked-tk NO) (checked-wall NO)(has-thickness NO))
         ?mycon <- (control (thickness-set ?))
         =>
                (printout writefile t crlf "          INFORMATION: Feature " ?num " is
of type " ?feat t crlf)
                (modify ?feat1 (feat-number ?num) (checked-wall YES))
         )
;;
;; NEED to count how many features have been read, and then test for when wall tkhasbeen
checked for all features.
```

```
;;
(defrule all-features-processed ""
        (not (feature (checked-wall NO)))
        (control (all-features-proc no))
        (part-design (feature-model yes))
          ?mycon <- (control (all-features-proc ?ap))
        =>
        (modify  ?mycon (all-features-proc yes))
        (printout t crlf "          ERRORCHECK: All the features have been processed"
        t crlf)
)

;;
;; Rule to define the design parameters if a feature model is not available
;;

(defrule define-design-params ""
       (control (thickness-set unset))
            ?mypart <- (part-design (part-name ?pname) (process ?proc1) (max-wall-tk
?maxw)(feature-model no))
         ?mycon <- (control (thickness-set ?tset))
            =>
         (printout
                 t crlf
                 "QUESTION: What is the maximum wall thickness? in mm "
                 t crlf)
          (bind ?val1 (read))
           (printout
                 t crlf
                 "QUESTION: What is the minimum wall thickness? in mm "
                 t crlf)
          (bind  ?val2 (read))
            (printout
                 t crlf
                 "QUESTION: What is the draft angle? in degrees "
                 t crlf)
          (bind  ?val3 (read))
         (modify ?mypart (max-wall-tk ?val1) (min-wall-tk ?val2) (draft-angle ?val3) )
         (modify ?mycon (thickness-set yes) (all-features-proc yes))
                (printout
                t crlf
                "          INFORMATION: The maximum wall thickness for the part has been
set to " ?val1 " mm"
                t crlf
                "          INFORMATION: The minimum wall thickness for the part has been
set to " ?val2 " mm"
                t crlf
                "          INFORMATION: The draft angle  for the part has been set to "
?val3 " degrees"
                t crlf )
                (printout writefile
                t crlf
                "          INFORMATION: The maximum wall thickness for the part has been
set to " ?val1 " mm"
                t crlf
                "          INFORMATION: The minimum wall thickness for the part has been
set to " ?val2 " mm"
                t crlf
                "          INFORMATION: The draft angle  for the part has been set to "
?val3 " degrees"
                t crlf )

         )

;;
;; If a material type has been selected offer possible processes
;;
(defrule select-process-type-plastic ""
       (part-design (part-name ?pname) (material-type plastic) (process unset))
         ?mypart <- (part-design (process ?proc1))

           =>
          (modify ?mypart (process injection-moulding))
          (printout
                 t crlf
                 "          INFORMATION: The process-type has been set to Injection
Moulding "
```

```
                 t crlf)
                (printout writefile
                 t crlf
                 "          INFORMATION: The process-type has been set to Injection
Moulding "
                 t crlf)

                 )
;;
;; If material is set to metal ask user to specify the manufacturing process
;;

(defrule select-process-type-metal  ""
       (part-design (part-name ?pname) (process unset) (material-type metal))
         ?mypart <- (part-design (part-name ?pname)(process ?proc))

         =>
         (printout
                  t crlf
                  "QUESTION: Is the manufacturing process SAND-CASTING, DIE-CASTING or
UNKNOWN?"
                  t crlf)
         (bind  ?newproc (read))
         (modify ?mypart (process ?newproc))
         (printout
                 t crlf
                 "          INFORMATION: The process-type has been set to " ?newproc
                 t crlf)
           (printout writefile
                 t crlf
                 "          INFORMATION: The process-type has been set to " ?newproc
                 t crlf)

         )

;;
;; Rule to set the process to  casting if material is metallic and process is unknown
;;

(defrule set-process-type-casting  ""
          (part-design (part-name ?pname) (process unknown) (material-type metal))
         ?mypart <- (part-design (part-name ?pname)(process ?proc))

         =>
         (printout
                  t crlf
                  "          INFORMATION: The manufacturing process has been set to
CASTING, final process selection will be based on the specicifed wall thickness"
                  t crlf)
         (modify ?mypart (process casting)))
;;
;; Rule to set the casting process based on the part thickness if user is unable to
specify a process
;;

 (defrule set-process-what-type-of-casting  ""
         (part-design (part-name ?pname) (process casting) (material-type metal) )
         (not (part-design (max-wall-tk 0.0)))
         ?mypart <- (part-design (part-name ?pname)(max-wall-tk ?max)(min-wall-tk ?min))
        ?sand-pars <- (process (proc-name sand-casting)(max-wall-tk ?maxsand) (min-wall-
tk ?minsand))
        ?die-pars <- (process (proc-name die-casting)(max-wall-tk ?maxdie) (min-wall-tk
?mindie))
          =>
         (if (> ?max ?maxsand )
             then
                 (printout writefile
                  t crlf
                  "         ADVICE: The wall thickness " ?max " is too thick to be
manufactured by a casting process "
                  t crlf))
          (if (and(and (> ?maxsand ?max) (> ?max ?minsand)) (> ?min ?minsand))
             then
                 (printout writefile
                  t crlf
                  "          ADVICE: The max wall thickness " ?max " and min wall
thickness " ?min " are appropriate for the sand-casting process "
```

```
                  t crlf
                  "          INFORMATION: Process has been set to SAND-CASTING " t crlf)
                  (modify ?mypart(process sand-casting)))
           (if  (and (and (> ?maxdie ?max) (> ?max ?mindie)) (> ?min ?mindie))
             then
                 (printout writefile
                  t crlf
                  "          ADVICE: The wall thickness " ?max " and min wall thckness "
?min " are appropriate for the die-casting process "
                  t crlf
                  "          INFORMATION: Process has been set to DIE-CASTING " t crlf)
                  (modify ?mypart (process die-casting)))
           (if (< ?min ?mindie)
               then
                 (printout writefile
                  t crlf
                  "          ADVICE: The wall thickness " ?min " is too thin to be
manufactured by a casting process "
                  t crlf))
         )
;;
;; If a process type has been set assign the correct materials
;;
(defrule select-material-type-inj-moulding ""
    (part-design (part-name ?pname) (process injection-moulding) (material-type unknown))
         ?mypart <- (part-design (material-type ?mat))
         =>
         (modify ?mypart (material-type plastic))
         (printout
                 t crlf
                 "          INFORMATION: The material type has been set to Plastic "
                 t crlf)
         )

(defrule select-process-casting  ""
        (part-design (part-name ?pname) (process sand-casting|die-casting) (material-
type unknown))
         ?mypart <- (part-design (material-type ?mat))

         =>
         (modify ?mypart (material-type metal))
         (printout
                 t crlf
                 "          INFORMATION: The material type has been set to Metal "
                 t crlf)
         )
;;
;; Obtain user specifications for the part to determine appropriate material and process
;;
;; Note at the moment this rule only checks whether strength is defined
;;
;;
(defrule collate-requirements ""
   (control (design-req unset) )
    (control (all-features-proc yes) )
  (not (part-design (material-type unknown|unset)))
   ?mypart <- (part-design (part-name ?pname))
   ?mycon <- (control (design-req ?req) (all-features-proc yes) )
        =>
                (printout
                t crlf
                "QUESTION: Do you want to define additional design requirements? "
                t crlf
                )
                (bind ?des-req (read))
                (if (eq ?des-req yes)then
                        (printout
                        t crlf
                        "QUESTION: Please enter a value between 0.0 and 1.0 for each of
the requirements listed below"
                        t crlf
                        "Where 0.0 is unimportant and 1.0 most important"
                        t crlf
                        "STRENGTH: ")
                        (bind  ?val5 (read))
                        (printout
                                t crlf
                                "APPEARANCE: ")
                        (bind  ?val6 (read))
                        (modify ?mypart (strength ?val5)(appearance ?val6))
                              (modify ?mycon (design-req yes))
                        (printout
                             t crlf
                             "          INFORMATION: The importance of STRENGTH has been
set to " ?val5 " on a scale of 0.0 to 1.0"
                             t crlf
                             "          INFORMATION: The importance of APPREARANCE has
been set to " ?val6 " on a scale of 0.0 to 1.0"
                             t crlf
                         )
                                (printout writefile
                             t crlf
                             "          INFORMATION: The importance of STRENGTH has been
set to " ?val5 " on a scale of 0.0 to 1.0"
                             t crlf
                             "          INFORMATION: The importance of APPREARANCE has
been set to " ?val6 " on a scale of 0.0 to 1.0"
                             t crlf
                         )

                    else
                        (modify ?mycon (design-req no))
                        )

        )
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Design Rules
;;
;; Wall Thickness - No appearance considerations
;;
;; GENR1
;;
(defrule wall-thickness-var  ""
         (not (control (design-req unset)))
         (control (mw-thickness-set yes)(thickness-set yes)(all-features-proc yes))
         (not (wall-tk-var-set ?))
         ?mypart <- (part-design (draft-angle ?angle2) (process ?proc1) (min-main-wall-
tk ?min1) (max-main-wall-tk ?max1)(appearance ?app&:(<= ?app 0.5)))
         =>
        (printout writefile
              t crlf
              t crlf
              "          ADVICE: Moulded parts should be designed with uniform wall
thickness (GENR1)"
              t crlf
         )
         (assert (wall-tk-var-set YES))
          (bind ?var  (abs (* (/ (- ?min1 ?max1) ?max1) 100)))
          (if (< ?var 30)
          then
         (printout writefile
          t crlf
         "          INFORMATION: The maximum main wall thickness on the part is " ?max1
" and the minimum main wall thickness is " ?min1
         "          ADVICE: The variation in the main-wall thickness on the part is "
?var " percent"
         t crlf
          "                 Which is acceptable for a part where appearance is not a
consideration (less than 0.5) "
          t crlf
         )
            else
         (printout writefile
         t crlf
         "          INFORMATION: The maximum main wall thickness on the part is " ?max1
" and the minimum main wall thickness is " ?min1
         "          ADVICE: The variation in the main-wall thickness for the part is "
?var " percent"
         t crlf
         "                 Which is NOT acceptable for a " ?proc1 " part, even if
appearance is not a consideration  "
```

```
         t crlf
         )
         )
)

;; Section change ratio
;;
;; GENR6
;;
(defrule section-change  ""
         (not (control (design-req unset)))
                (control (all-features-proc yes))
          (wall-tk-var-set ?)
         ?mypart <- (part-design (process ?proc1)(draft-angle ?angle2) (min-wall-tk
?min1) (max-wall-tk ?max1) (appearance ?app))
         (process (proc-name ?proc1) (section-change-ratio ?scr))

         =>
          (bind ?var  (abs (* (/ (- ?min1 ?max1) ?max1) 100)))
          (if (> ?var 10)
          then
              (printout writefile
              t crlf
              "          ADVICE: The design should not have abrupt section changes. "
              t crlf
              "                   Where a section change is required a gradual taper of
" ?scr " must be applied (GENR6)"
              t crlf
              )

          )
)

;; Corner Radii
;;
;; GENR7
;;
(defrule corner-radii  ""
         (not (control (design-req unset)))
           (control (all-features-proc yes))
         (wall-tk-var-set ?)
         ?mypart <- (part-design (draft-angle ?angle2) (min-wall-tk ?min1) (max-wall-tk
?max1) (appearance ?app))

         =>
              (printout writefile
              t crlf
              "          ADVICE: All Corners should be generously radiussed (GENR7)"
              t crlf

              )
)

;;
;; Maximum/ Minimum Wall Thickness
;;
;; GENR2
;;

 (defrule max-wall-thickness-process
         (not (part-design (process casting|unset)))
           (control (all-features-proc yes))
         ?mypart <- (part-design (process ?proc1)(material-type ?mat1) (material-name
unset|unknown) (max-wall-tk  ?max)(min-wall-tk ?min) )
        (process (proc-name ?proc1) (max-wall-tk ?maxwalltk) (min-wall-tk ?minwalltk))

         =>
         (if (and (< ?max ?maxwalltk) (> ?max ?minwalltk))
         then
             (modify ?mypart (too-thick no))
             (printout writefile
                    t crlf
                    "          ADVICE: The specified maximum wall thickness " ?max " is
acceptable for selected process " ?proc1
                    "                  (maximum thickness for " ?proc1 " is " ?maxwalltk
 " (GENR2) "
```

```
                    t crlf      )

          else
              (if (> ?max ?maxwalltk)
              then
;;              (modify ?mypart (too-thick yes))
              (printout writefile
                           t crlf
                           "          ADVICE: The specificed max wall thickness is "
?max " which is greater than "
                           t crlf
                           "                  the maximum wall thickness for  " ?proc1 "
( " ?maxwalltk " mm.)"
                           t crlf
                           )
                  else
                  (printout writefile
                           t crlf
                           "          ADVICE: The specificed max wall thickness  " ?max
"  is less than "
                           t crlf
                           "                  the minimum wall thickness for a " ?proc1
" part (" ?minwalltk " mm.)"
                           t crlf
                           "                  you are recommended to increase the wall
thickness or select an alternative material with improved mechanical properties (GENR4)"
                           t crlf
                           ))
                           )
                           )


(defrule min-wall-thickness-process
         (not (part-design (process casting|unset)))
                (control (all-features-proc yes))
         ?mypart <- (part-design (process ?proc1)(material-type ?mat1) (material-name
unset|unknown) (max-wall-tk  ?max)(min-wall-tk ?min) )
        (process (proc-name ?proc1) (max-wall-tk ?maxwalltk) (min-wall-tk ?minwalltk))

         =>
         (if (and (> ?min ?minwalltk) (< ?min ?maxwalltk))
         then
             (printout writefile
                    t crlf
                    "          ADVICE: The specified minimum wall thickness " ?min " is
acceptable for selected process " ?proc1
                    "                  (the minimum thickness for process " ?proc1 " is "
?minwalltk  " )"
                    t crlf
                  )

          else
              (if (> ?min ?maxwalltk)
                  then
                  (printout writefile
                           t crlf
                           "          ADVICE: The specificed min wall thickness  " ?min
"  is greater than "
                           t crlf
                           "                  the maximum wall thickness for a " ?proc1
" part (" ?maxwalltk " mm.)"
                           t crlf
                           " you are recommended to reduce the minimum wall thickness "
                           t crlf
                           )
                  else
                  (printout writefile
                           t crlf
                           "          ADVICE: The specificed min wall thickness  " ?min
"  is less than "
                           t crlf
                           "                  the minimum wall thickness for a " ?proc1
" part (" ?minwalltk " mm.)"
                           t crlf
                           "                  you are recommended to increase the
minimum wall thickness "
```

```
                           t crlf
                           ))
         )
 )

 (defrule max-wall-thickness-material
         (not (part-design (process casting|unset)))
           (control (all-features-proc yes))
         (not (part-design (material-name unset)))
         ?mypart <- (part-design (process ?proc1)(material-type ?mat1) (material-name
?mat2) (max-wall-tk  ?max)(min-wall-tk ?min) )
        (material (material-name ?mat2) (process-name ?proc1) (max-wall-tk ?maxwalltk)
(min-wall-tk ?minwalltk))
        (not (max-ckd ?))

         =>
            (assert (max-ckd yes))
         (if (and (< ?max ?maxwalltk) (> ?max ?minwalltk))
         then
             (modify ?mypart (too-thick no))
             (printout writefile
                    t crlf
                    "         ADVICE: The specified maximum wall thickness " ?max " is
acceptable for selected "
                    t crlf
                    "                 material " ?mat2  " and process " ?proc1 "
(maximum thickness for process is " ?maxwalltk  " ) (GENR2)"
                    t crlf      )

          else
              (if (> ?max ?maxwalltk)
              then
              (modify ?mypart (too-thick yes))
              (printout writefile
                           t crlf
                           "          ADVICE: The specificed max wall thickness is "
?max " which is greater than "
                           t crlf
                           "                  the maximum wall thickness for a " ?mat2 "
part manufactured using " ?proc1 "  (" ?maxwalltk " mm.) "
                           t crlf
                           )
                  else
                  (printout writefile
                           t crlf
                           "          ADVICE: The specificed max wall thickness  " ?max
"  is less than "
                           t crlf
                           "                  the minimum wall thickness for a " ?mat2 "
part manufactured using " ?proc1 "  (" ?minwalltk " mm.)"
                           t crlf
                           "                  you are recommended to increase the
minimum wall thickness to at least " ?minwalltk " mm. "
                           t crlf
                           ))
                           )
                           )


(defrule min-wall-thickness-material
         (not (part-design (process casting|unset)))
                (control (all-features-proc yes))
         (not (part-design (material-name unset)))
         ?mypart <- (part-design (process ?proc1)(material-type ?mat1) (material-name
?mat2) (max-wall-tk  ?max)(min-wall-tk ?min) )
        (material (material-name ?mat2) (process-name ?proc1) (max-wall-tk ?maxwalltk)
(min-wall-tk ?minwalltk))
        (not (min-ckd ?))

         =>
           (assert (min-ckd yes))
         (if (and (> ?min ?minwalltk) (< ?min ?maxwalltk))
         then
             (printout writefile
                    t crlf
                    "          ADVICE: The specified minimum wall thickness " ?min " is
acceptable for selected material " ?mat2 " and process " ?proc1  ". "
```

```
                    t crlf
                    "                  (minimum thickness for material is " ?minwalltk
")"
                    t crlf
                  )

          else
              (if (> ?min ?maxwalltk)
                  then
                  (printout writefile
                           t crlf
                           "          ADVICE: The specificed min wall thickness  " ?min
"  is greater than "
                           t crlf
                           "                  the maximum wall thickness for a " ?mat2 "
part manufactured using " ?proc1 " process"
                           t crlf
                           "                  you are recommended to reduce the minimum
wall thickness to less than " ?maxwalltk " mm (GENR2)"
                           t crlf
                           )
                  else
                  (printout writefile
                           t crlf
                           "          ADVICE: The specificed min wall thickness  " ?min
"  is less than "
                           t crlf
                           "                  the minimum wall thickness for a " ?mat2 "
part "
                           t crlf
                           "                  you are recommended to increase the
minimum wall thickness to more than " ?minwalltk " mm "
                           t crlf
                           ))
         )
 )

 ;; Wall for minimum draft angle
 ;;
 ;; GENR5
 ;;
 (defrule min-draft-angle
         (not (part-design (process casting|unset)))
                (control (all-features-proc yes))
          ?mypart <- (part-design (process ?proc1)(draft-angle ?draft))
        (process (proc-name ?proc1) (min-draft-angle ?mindraft))
        (not (draft-ckd ?))

         =>
           (assert (draft-ckd yes))
           (if (< ?draft ?mindraft)
                  then
                  (printout writefile
                           t crlf
                           "          ADVICE: The specificed draft angle  " ?draft "
degrees is less than "
                           t crlf
                           "                  the minimum draft angle for a part
manufactured using " ?proc1
                           t crlf
                           "                  you are recommended to reduce the increase
the draft angle to at least " ?mindraft " degrees (GENR5)"
                           t crlf
                           )
                  else
                  (printout writefile
                           t crlf
                           "          ADVICE: The specificed draft angle of " ?draft "
degrees is acceptable for a part manufactured using " ?proc1 ". (GENR5) "
                           t crlf
                           )
            )
 )

  ;;
;; Wall thickness with strength considerations
```

```
;;
;; GENR3, GENR4
;;
;;
(defrule wall-thickness-strength
      (not (control (design-req unset)))
      (part-design (too-thick yes))
      ?mypart <- (part-design (process ?proc1)(strength  ?str)(max-wall-tk ?tk))
         (process (proc-name ?proc1) (max-wall-tk ?maxwalltk))
          (not (wall-tk-strength ?))
         =>
         (assert (wall-tk-strength YES))
         (if (> ?str 0.3 )
         then
             (printout writefile
                  t crlf
                  "          ADVICE: Strength has been input as important (" ?str "). "
                    t crlf
                  "                  In order to meet the STRENGTH requirement with a
smaller wall thickness "
                  t crlf
                  "                  it is recommended that you redesign the part using
ribs or corrugations (GENR2, GENR4)"
                  )

          else
              (printout writefile
                  t crlf
                  "          ADVICE: Strength is a low considerarion (" ?str "). "
                    t crlf
                  "                  it is recommended that you redesign the part with a
smaller wall thickness (GENR2)"
                  ))
         )

;;
;; Wall thickness variation, appearance considerations
;;
;; GENR8
;;

(defrule wall-thickness-var-app  ""
         (not (control (design-req unset)))
         (part-design (process injection-moulding))
         (not (wall-tk-var-set ?))
         ?mypart <- (part-design (draft-angle ?angle2) (min-main-wall-tk ?min1) (max-
main-wall-tk ?max1) (appearance ?app&:(> ?app 0.5)))
         =>
         (printout writefile
                    t crlf
                      "          ADVICE: Moulded parts should be designed with uniform
wall thickness (GENR1)"
                    t crlf
                     )
         (assert (wall-tk-var-set YES))
          (bind ?var  (abs (* (/ (- ?min1 ?max1) ?max1) 100)))
         (if (< ?var 10)
             then
             (printout writefile
             t crlf
             "          ADVICE: Variations in wall thickness should be minimised for
parts in which appearance is important"
             t crlf
             "                  The variation in main wall thickness is " ?var " percent"
             t crlf
              "                 Which is acceptable for a part with high importance for
appearance (more than 0.5) (IMR3)"
              t crlf
             )
             else
                 (printout writefile
                 t crlf
                 "          ADVICE: The variation in main wall thickness is " ?var "
percent"
                 t crlf
                 "                  Which is NOT acceptable for a part with high
importance for appearance (more than 0.5) "
```

```
                 t crlf
                  "                 If wall thickness variation is essential then
recommend use of textured surface"
                  t crlf
                  "                 and increase draft angle by 1 degree " )

 ;;       (modify ?des1 (draft-angle (+ ?angle2 1))(min-wall-tk ?min1) (max-wall-tk
?max1) (initialise YES))
         ))

;;
;; Rule for Boss design - Die casting
;;
;; DCR1
;;
    (defrule boss-thickness-die ""
         (part-design (process die-casting))
         (control (set-default-wall yes))
         (wall-tk-var-set YES)
         ?model1 <- (feature-model (thickness ?wall-tk))
         ?feat1 <- (feature (feat-type boss) (feat-number ?num) (thickness ?boss-tk)
(checked-tk NO))
         ?proc1 <- (process (proc-name die-casting)(rec-buttress-prop ?rec-buttress-
prop1))
             =>
;;              (printout
;;                      t crlf
;;                      "          ERROR CHECK: value of proc-name is " ?proctype "" t
crlf
;;                      "          ERROR CHECK: value of rec-buttress-prop is " ?rec-
buttress-prop1 "" t crlf
;;                      "          ERROR CHECK: Value of wall-tk is " ?wall-tk "" t crlf
;;                      "          ERROR CHECK: Value of boss-tk is " ?boss-tk "" t crlf
;;
;;                      )
         (bind ?boss-prop (*(/ ?boss-tk ?wall-tk ) 100))

         (printout writefile
                  t crlf
                  "          ADVICE: (Feature " ?num ") Projections and bosses can be
difficult to fill:"  t crlf
                  "                  buttresses assist flow of such features and
strengthen the component (DCR1)"
                  t crlf
                  )
                  (bind ?rec-tk (* ?wall-tk ?rec-buttress-prop1))
                  (if  (> ?boss-tk  ?rec-tk)
                       then
                           (printout writefile
                           t crlf
                           "          ADVICE: (Feature " ?num ") boss is too thick and
should be reduced to " ?rec-tk   ""
                           t crlf
                           )
                  else
                          (printout writefile
                           t crlf
                           "          ADVICE: (Feature " ?num ") boss is of acceptable
thickness "
                           t crlf
                           )
                           )
(modify ?feat1 (checked-tk YES))
)
;;
;; Rib Design -Die Casting
;;
;; DCR2,3
;;
(defrule rib-design-die ""
         (part-design (process die-casting))
 ;;        (not (part-design (process casting|unset)))
         (control(set-default-wall yes))
         (wall-tk-var-set YES)
         ?procname <- (process (proc-name die-casting)(rec-rib-prop ?rec-rib-prop1))
         ?model1 <- (feature-model (thickness ?wall-tk))
```

```
         ?feat1 <- (feature (feat-type rib) (feat-number ?num) (thickness ?rib-tk)
(checked-tk NO))
             =>
;;              (printout
;;                      t crlf
;;                      "          ERROR CHECK: value of rec-rib-prop is " ?rec-rib-
prop1 "" t crlf
;;                      "          ERROR CHECK: Value of wall-tk is " ?wall-tk "" t crlf
;;                      "          ERROR CHECK: Value of rib-tk is " ?rib-tk "" t crlf
;;                      )
         (bind ?rib-prop (*(/ ?rib-tk ?wall-tk ) 100))
         (bind ?rec-rib-prop-pc (* ?rec-rib-prop1 100))
         (printout writefile
                  t crlf
                  "          ADVICE: (Feature " ?num ") rib. Ribs should not be square
in section."
                  t crlf
                  "                   Blended sections and curves buttresses aid die
filling (DCR2)"
                  t crlf
                  "          ADVICE: (Feature " ?num ") rib has thickness  " ?rib-tk
                  " mm which is " ?rib-prop " percent of the main wall thickness ("
?wall-tk " mm )"
                  t crlf
                  "          ADVICE: For process die-casting the recommended rib
thickness is "   ?rec-rib-prop-pc " percent of main wall thickness"
                  )
                  (bind ?rec-tk (* ?wall-tk ?rec-rib-prop1))
                  (if  (> ?rib-tk  ?rec-tk)
                       then
                           (printout writefile
                           t crlf
                           "          ADVICE: (Feature " ?num ") rib is too thick and
should be reduced to " ?rec-tk   ""
                           t crlf
                           "                   You are recommended to avoid thick
sections atintersections of ribs (DCR3)"
                           t crlf
                           )
                  else
                          (printout writefile
                           t crlf
                           "          ADVICE: (Feature " ?num ") rib is of acceptable
thickness "
                           t crlf
                           )
                           )
(modify ?feat1 (checked-tk YES))                           )

;;
;; Hole Design - Die Casting
;;
;; DCR4, DCR5
;;

(defrule hole-design-die ""
         (part-design (process die-casting))
         (control (set-default-wall yes))
         (wall-tk-var-set YES)
         ?procname <- (process (proc-name die-casting))
         ?feat1 <- (feature (feat-type hole) (feat-number ?num))
             =>
             (printout writefile
                        t crlf
                        "          ADVICE: (Feature " ?num ") Blind holes are preferable
to through holes. "
                        t crlf
                        "                   Through holes can cause problems with flash.
(DCR4) "
                         t crlf
                        "          ADVICE: (Feature " ?num ") Holes should be tapered "
                        t crlf
                        "                   Tapered holes assist with removal of the
casting from the die. (DCR5) "
                        t crlf )
)
```

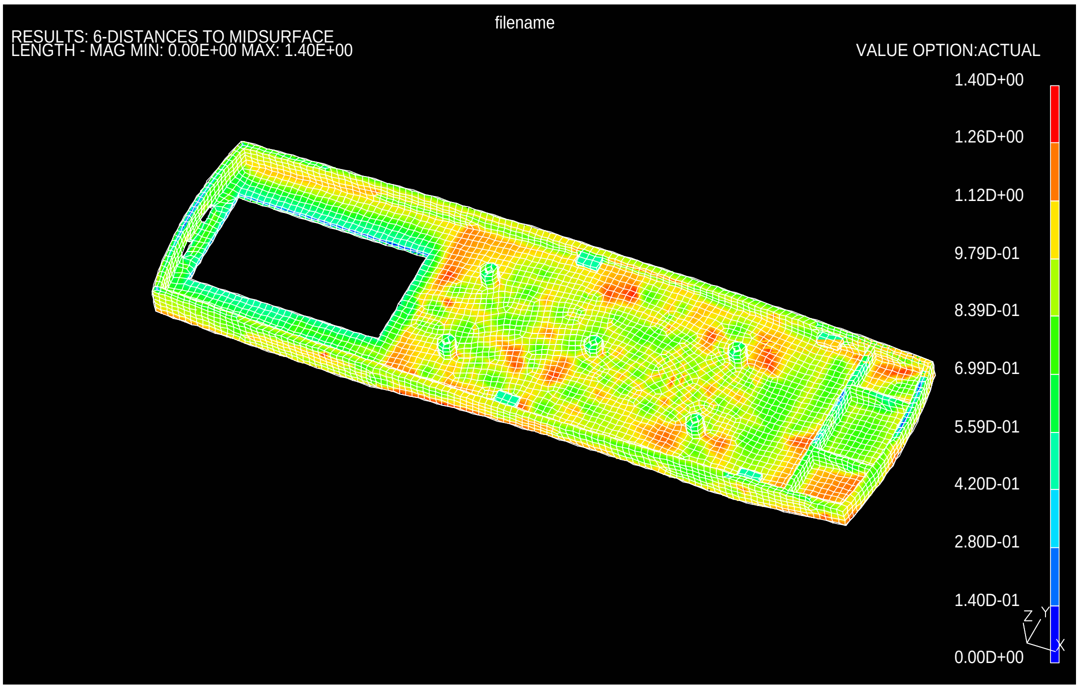
```
;;
;; Rib Design -Injection Moulding
;;
;; IMR1
;;
(defrule rib-design ""
         (part-design (process injection-moulding))
         (not (part-design (process casting|unset)))
         (control(set-default-wall yes))
         (wall-tk-var-set YES)
         ?procname <- (process (proc-name injection-moulding)(rec-rib-prop ?rec-rib-
prop1))
         ?model1 <- (feature-model (thickness ?wall-tk))
         ?feat1 <- (feature (feat-type rib) (feat-number ?num) (thickness ?rib-tk)
(checked-tk NO))
             =>
;;              (printout
;;                      t crlf
;;                      "          ERROR CHECK: value of rec-rib-prop is " ?rec-rib-
prop1 "" t crlf
;;                      "          ERROR CHECK: Value of wall-tk is " ?wall-tk "" t crlf
;;                      "          ERROR CHECK: Value of rib-tk is " ?rib-tk "" t crlf
;;                      )
         (bind ?rib-prop (*(/ ?rib-tk ?wall-tk ) 100))
          (bind ?rec-rib-prop-pc (* ?rec-rib-prop1 100))
         (printout writefile
                  t crlf
                  "          INFORMATION: Rib " ?num " has thickness  " ?rib-tk
                  " mm which is " ?rib-prop " percent of the main wall thickness ("
?wall-tk " mm )"
                  t crlf
                  "          ADVICE: For process injection-moulding the recommended rib
thickness is "   ?rec-rib-prop-pc " percent of main wall thickness"
                  )
                  (bind ?rec-tk (* ?wall-tk ?rec-rib-prop1))
                  (if  (> ?rib-tk  ?rec-tk)
                       then
                           (printout writefile
                           t crlf
                           "                  Rib " ?num " is too thick and should be
reduced to " ?rec-tk   ""
                           t crlf
                           )
                  else
                          (printout writefile
                           t crlf
                           "                  Rib " ?num " is of acceptable thickness "
                           t crlf
                           )
                           )
(modify ?feat1 (checked-tk YES))

                           )
   (defrule buttress-thickness ""
         (process-type ?proctype)
         (wall-tk-var-set YES)
         (not (part-design (process casting|unset)))
           ?procname <- (process (proc-name ?proctype)(rec-buttress-prop ?rec-buttress-
prop1))
         ?model1 <- (feature-model (thickness ?wall-tk))
         ?feat1 <- (feature (feat-type buttress) (feat-number ?num) (thickness ?butt-tk)
(checked-tk NO))
             =>
;;              (printout
;;                      t crlf
;;                      "          ERROR CHECKvalue of proc-name is " ?proctype "" t crlf
;;                      "          ERROR CHECKvalue of rec-buttress-prop is " ?rec-
buttress-prop1 "" t crlf
;;                      "          ERROR CHECKValue of wall-tk is " ?wall-tk "" t crlf
;;                      "          ERROR CHECKValue of butt-tk is " ?butt-tk "" t crlf
;;
;;                      )
         (bind ?butt-prop (*(/ ?butt-tk ?wall-tk ) 100))

         (printout writefile
```

```
                  t crlf
                  "          INFORMATION: The buttress thickness is " ?butt-tk
                  "                  mm and the wall thickness is " ?wall-tk " mm " t
crlf "The buttress thickness is "
                  "                  "?butt-prop " percent of the main wall thickness "
                  t crlf
                  "                  For process "?proctype " the recommended buttress
thickness is "   ?rec-buttress-prop1 " x main wall thickness"
                  t crlf)
                  (bind ?rec-tk (* ?wall-tk ?rec-buttress-prop1))
                  (if  (> ?butt-tk  ?rec-tk)
                       then
                           (printout writefile
                           t crlf
                           "          ADVICE: The buttress " ?num " is too thick and
should be reduced to " ?rec-tk   ""
                           t crlf
                           )
                  else
                          (printout writefile
                           t crlf
                           "          ADVICE: The buttress " ?num " is of acceptable
thickness "
                           t crlf
                           )
                           )
(modify ?feat1 (checked-tk YES))

                           )

(defrule close-file ""
        (control (design-req yes|no)(thickness-set yes) (all-features-proc yes))
        =>
        (printout
        t crlf
                 "           ERRORCHECK: ALL FINISHED - closing file for writing"
        t crlf)
        (close writefile)
)
```

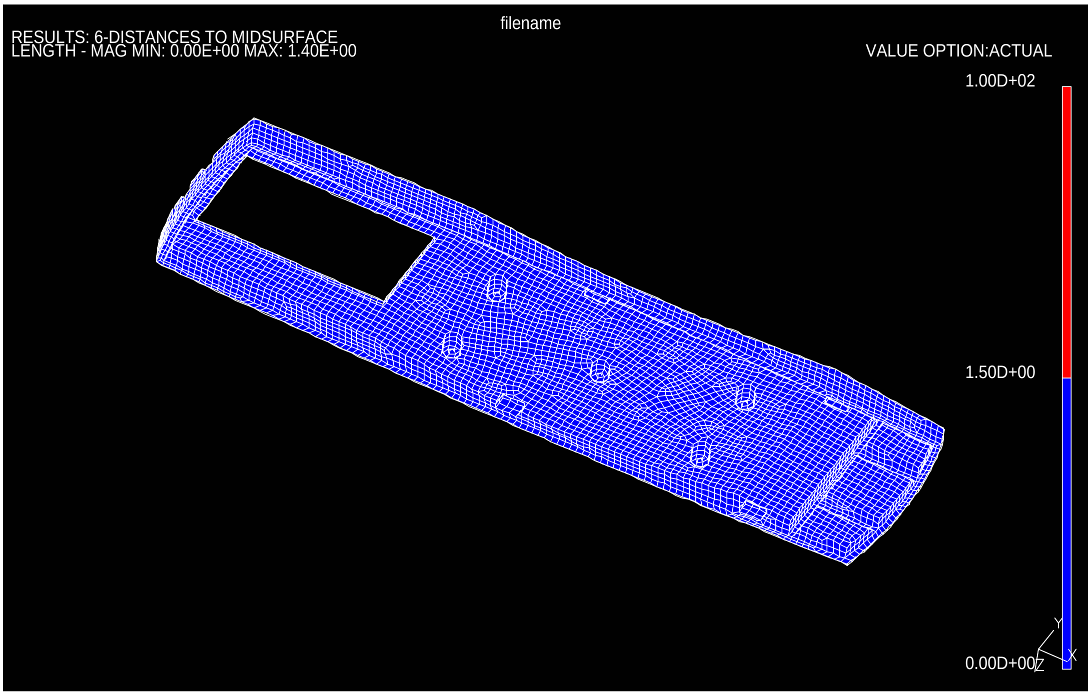
APPENDIX D - Test Case Results

Remote Control Case Study – Results

1. Mid-surface Quality Evaluation



Contour Plot of Hausdorff Distances (maximum distance from solid model to mid-surface model = 1.4 mm, grid spacing = 2.0 mm)



Contour Plot of Hausdorff Distances (Threshold set at 10% over wall thickness, number of failed points = 0 out of 10,009)

2. Feature Recognition Results Evaluation

Feature recognition results:
5 ribs, 5 bosses, 4 fins, 1 hole and 19 main wall faces.

Feature recognition results evaluation:

F_{solid} = (19 + 8 – 0) + (2*14 + 21 - 2)
= 74

Number of Faces in original solid part (F_{actual}):
83

FRF = F_{solid}/ F_{actual} = 0.89

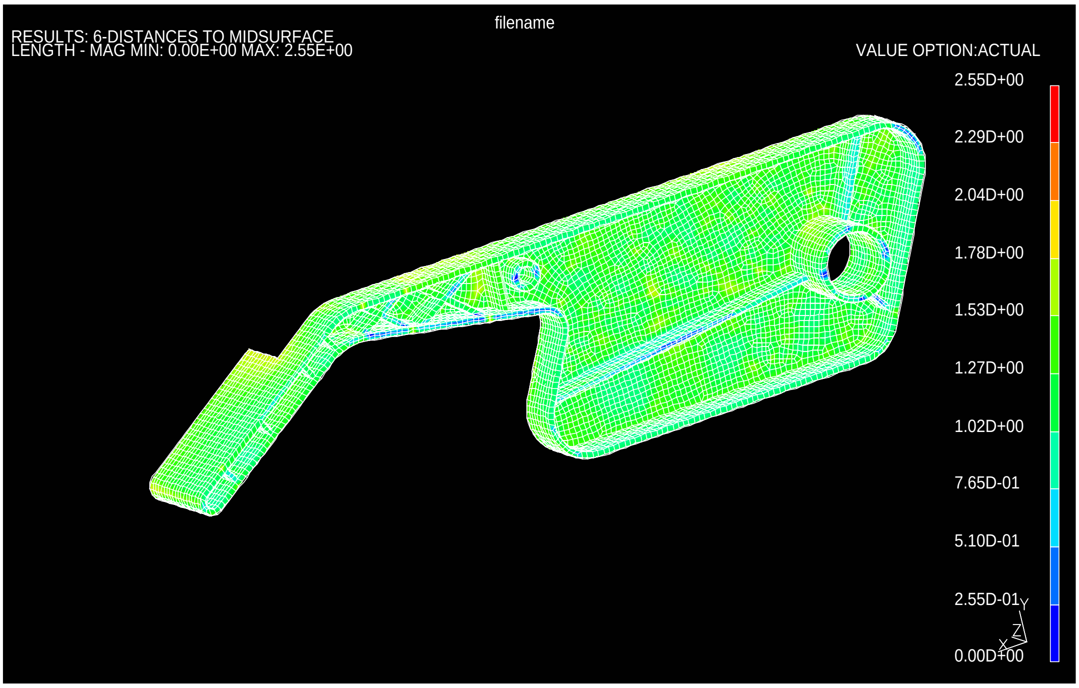
3. Manufacturing Advisor Report File:

```
 ************** OUTPUT FROM KNOWLEDGE BASED MOULDING ADVISOR*****************

 **********************WRITTEN BY HELEN LOCKETT*******************************

 **************MANUFACUTRING ADVICE FOR PART Remote-Control*******************

INFORMATION: The material has been set to acrylic
INFORMATION: The process-type has been set to Injection Moulding
INFORMATION: A feature model has been read from the file remote-feat-file2.txt
INFORMATION: The nominal wall thickness has been set to 1.5 (based on value
from feature file)
INFORMATION: Feature 34 is of type main wall and has thickness 1.5  mm
INFORMATION: Feature 16 is of type main wall and has thickness 1.5  mm
INFORMATION: Feature 19 is of type main wall and has thickness 1.5  mm
INFORMATION: Feature 20 is of type main wall and has thickness 1.5  mm
INFORMATION: Feature 21 is of type main wall and has thickness 1.5  mm
INFORMATION: Feature 23 is of type main wall and has thickness 1.5  mm
INFORMATION: Feature 25 is of type main wall and has thickness 1.5  mm
INFORMATION: Feature 26 is of type main wall and has thickness 1.5  mm
INFORMATION: Feature 28 is of type main wall and has thickness 1.5  mm
INFORMATION: Feature 29 is of type main wall and has thickness 1.5  mm
INFORMATION: Feature 30 is of type main wall and has thickness 1.5  mm
INFORMATION: Feature 31 is of type main wall and has thickness 1.5  mm
INFORMATION: Feature 32 is of type main wall and has thickness 1.5  mm
INFORMATION: Feature 33 is of type main wall and has thickness 1.5  mm
INFORMATION: The importance of STRENGTH has been set to 0.3 on a scale of 0.0
to 1.0
INFORMATION: The importance of APPREARANCE has been set to 0.9 on a scale of
0.0 to 1.0
ADVICE: The specified maximum wall thickness 1.5 is acceptable for selected
material acrylic and process injection-moulding (maximum thickness for process
is 3.0 ) (GENR2)
ADVICE: The specified minimum wall thickness 1.0 is acceptable for selected
material acrylic and process injection-moulding.
 (minimum thickness for material is 0.6)
ADVICE: The specificed draft angle  0.0  degrees is less than
the minimum draft angle for a part manufactured using injection-moulding
you are recommended to reduce the increase the draft angle to at least 0.5
degrees (GENR5)
ADVICE: Moulded parts should be designed with uniform wall thickness (GENR1)
ADVICE: Variations in wall thickness should be minimised for parts in which
appearance is important
The variation in main wall thickness is 0.0 percent
Which is acceptable for a part with high importance for appearance (more than
ADVICE: The design should not have abrupt section changes.
Where a section change is required a gradual taper of 3.0 must be applied
(GENR6)
ADVICE: All Corners should be generously radiussed (GENR7)
```

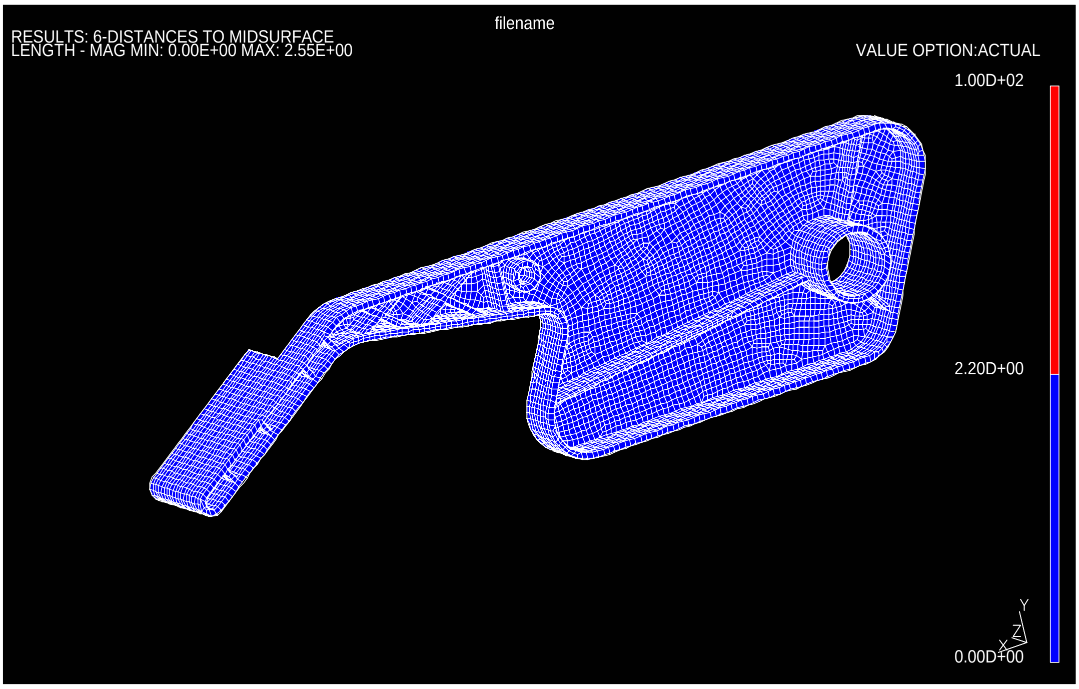
```
INFORMATION: Rib 27 has thickness  1.0 mm which is 66.6666666666667 percent of
the main wall thickness (1.5 mm )
ADVICE: For process injection-moulding the recommended rib thickness is 60.0
percent of main wall thickness
Rib 27 is too thick and should be reduced to 0.9
INFORMATION: Rib 24 has thickness  1.0 mm which is 66.6666666666667 percent of
the main wall thickness (1.5 mm )
ADVICE: For process injection-moulding the recommended rib thickness is 60.0
percent of main wall thickness
Rib 24 is too thick and should be reduced to 0.9
INFORMATION: Rib 22 has thickness  1.5 mm which is 100.0 percent of the main
wall thickness (1.5 mm )
ADVICE: For process injection-moulding the recommended rib thickness is 60.0
percent of main wall thickness
Rib 22 is too thick and should be reduced to 0.9
INFORMATION: Rib 18 has thickness  1.5 mm which is 100.0 percent of the main
wall thickness (1.5 mm )
ADVICE: For process injection-moulding the recommended rib thickness is 60.0
percent of main wall thickness
Rib 18 is too thick and should be reduced to 0.9
INFORMATION: Rib 17 has thickness  1.0 mm which is 66.6666666666667 percent of
the main wall thickness (1.5 mm )
ADVICE: For process injection-moulding the recommended rib thickness is 60.0
percent of main wall thickness
Rib 17 is too thick and should be reduced to 0.9
```

Seat Adjuster Case Study – Results

1. Mid-surface Quality Evaluation (edited version)



Contour Plot of Hausdorff Distances (maximum distance from solid model to mid-surface model = 2.55 mm, grid spacing = 2.0 mm)


filename
RESULTS: 6-DISTANCES TO MIDSURFACE
LENGTH - MAG MIN: 0.00E+00 MAX: 2.55E+00
VALUE OPTION:ACTUAL
1.00D+02
2.20D+00
0.00D+00
Y
Z
X

Contour Plot of Hausdorff Distances (threshold set at 10% over wall thickness, number of failed points = 4 out of 16238)

2. Feature Recognition Results Evaluation (unedited version)

Feature recognition results:
6 ribs, 13 buttresses, 1 boss and 55 main wall faces.

Feature recognition results evaluation:

F_{solid} = (55 + 18 – 12) + (2*20 + 24 - 0)
= 125

Number of Faces in original solid part (F_{actual}):
102 FRF = F_{solid}/ F_{actual} = 1.23

3. Feature Recognition Results Evaluation (edited version)

Feature recognition results:
14 ribs, 2 buttresses, 1 boss and 53 main wall faces.

Feature recognition results evaluation:

F_{solid} = (53 + 18 – 12) + (2*17 + 14 - 1)
= 106

Number of Faces in original solid part (F_{actual}):
102 FRF = F_{solid}/ F_{actual} = 1.04

4. Manufacturing Advisor Report File (edited version):

```
**************** OUTPUT FROM KNOWLEDGE BASED MOULDING ADVISOR****************

************************WRITTEN BY HELEN LOCKETT*****************************

       ************MANUFACUTRING ADVICE FOR PART Seat-Adjuster***************


INFORMATION: The material has been set to acetal
INFORMATION: The process-type has been set to Injection Moulding
INFORMATION: A feature model has been read from the file seatadj-fea-file.txt
INFORMATION: The nominal wall thickness has been set to 2.0 (based on value
from feature file)
INFORMATION: Feature 69 is of type main wall and has thickness 2.0  mm
INFORMATION: Feature 2 is of type main wall and has thickness 2.0  mm
INFORMATION: Feature 3 is of type main wall and has thickness 2.0  mm
INFORMATION: Feature 4 is of type main wall and has thickness 2.0  mm
INFORMATION: Feature 5 is of type main wall and has thickness 2.0  mm
INFORMATION: Feature 7 is of type main wall and has thickness 2.0  mm
INFORMATION: Feature 8 is of type main wall and has thickness 2.0  mm
INFORMATION: Feature 10 is of type main wall and has thickness 2.0  mm
INFORMATION: Feature 11 is of type main wall and has thickness 2.0  mm
INFORMATION: Feature 12 is of type main wall and has thickness 2.0  mm
INFORMATION: Feature 13 is of type main wall and has thickness 2.0  mm
INFORMATION: Feature 14 is of type main wall and has thickness 2.0  mm
INFORMATION: Feature 16 is of type main wall and has thickness 2.0  mm
INFORMATION: Feature 18 is of type main wall and has thickness 2.0  mm
INFORMATION: Feature 19 is of type main wall and has thickness 2.0  mm
```

```
INFORMATION: Feature 20 is of type main wall and has thickness 2.0  mm
INFORMATION: Feature 21 is of type main wall and has thickness 2.0  mm
INFORMATION: Feature 22 is of type main wall and has thickness 2.0  mm
INFORMATION: Feature 23 is of type main wall and has thickness 2.0  mm
INFORMATION: Feature 24 is of type main wall and has thickness 2.0  mm
INFORMATION: Feature 25 is of type main wall and has thickness 2.0  mm
INFORMATION: Feature 26 is of type main wall and has thickness 2.0  mm
INFORMATION: Feature 27 is of type main wall and has thickness 2.0  mm
INFORMATION: Feature 28 is of type main wall and has thickness 2.0  mm
INFORMATION: Feature 30 is of type main wall and has thickness 2.0  mm
INFORMATION: Feature 32 is of type main wall and has thickness 2.0  mm
INFORMATION: Feature 34 is of type main wall and has thickness 2.0  mm
INFORMATION: Feature 35 is of type main wall and has thickness 2.0  mm
INFORMATION: Feature 37 is of type main wall and has thickness 2.0  mm
INFORMATION: Feature 38 is of type main wall and has thickness 2.0  mm
INFORMATION: Feature 39 is of type main wall and has thickness 2.0  mm
INFORMATION: Feature 40 is of type main wall and has thickness 2.0  mm
INFORMATION: Feature 41 is of type main wall and has thickness 2.0  mm
INFORMATION: Feature 46 is of type main wall and has thickness 2.0  mm
INFORMATION: Feature 47 is of type main wall and has thickness 2.0  mm
INFORMATION: Feature 48 is of type main wall and has thickness 2.0  mm
INFORMATION: Feature 49 is of type main wall and has thickness 2.0  mm
INFORMATION: Feature 50 is of type main wall and has thickness 2.0  mm
INFORMATION: Feature 51 is of type main wall and has thickness 2.0  mm
INFORMATION: Feature 52 is of type main wall and has thickness 2.0  mm
INFORMATION: Feature 53 is of type main wall and has thickness 2.0  mm
INFORMATION: Feature 54 is of type main wall and has thickness 2.0  mm
INFORMATION: Feature 56 is of type main wall and has thickness 2.0  mm
INFORMATION: Feature 57 is of type main wall and has thickness 2.0  mm
INFORMATION: Feature 58 is of type main wall and has thickness 2.0  mm
INFORMATION: Feature 60 is of type main wall and has thickness 2.0  mm
INFORMATION: Feature 61 is of type main wall and has thickness 2.0  mm
INFORMATION: Feature 62 is of type main wall and has thickness 2.0  mm
INFORMATION: Feature 64 is of type main wall and has thickness 2.0  mm
INFORMATION: Feature 65 is of type main wall and has thickness 2.0  mm
INFORMATION: Feature 66 is of type main wall and has thickness 2.0  mm
INFORMATION: Feature 68 is of type main wall and has thickness 2.0  mm
INFORMATION: Feature 1 is of type boss and has thickness 2.0  mm
INFORMATION: Feature 6 is of type buttress and has thickness 2.0  mm
INFORMATION: Feature 9 is of type rib and has thickness 1.5  mm
INFORMATION: Feature 15 is of type rib and has thickness 2.0  mm
INFORMATION: Feature 17 is of type rib and has thickness 1.5  mm
INFORMATION: Feature 29 is of type rib and has thickness 1.5  mm
INFORMATION: Feature 31 is of type buttress and has thickness 1.5  mm
INFORMATION: Feature 33 is of type rib and has thickness 1.5  mm
INFORMATION: Feature 36 is of type rib and has thickness 2.0  mm
INFORMATION: Feature 42 is of type rib and has thickness 2.0  mm
INFORMATION: Feature 43 is of type rib and has thickness 2.0  mm
INFORMATION: Feature 44 is of type rib and has thickness 2.0  mm
INFORMATION: Feature 45 is of type rib and has thickness 2.0  mm
INFORMATION: Feature 55 is of type rib and has thickness 1.5  mm
INFORMATION: Feature 59 is of type rib and has thickness 1.5  mm
INFORMATION: Feature 63 is of type rib and has thickness 1.5  mm
INFORMATION: Feature 67 is of type rib and has thickness 1.5  mm
INFORMATION: Feature 69 is of type main-wall and has thickness 2.0  mm
INFORMATION: The importance of STRENGTH has been set to 1.0 on a scale of 0.0
to 1.0
INFORMATION: The importance of APPREARANCE has been set to 0.0 on a scale of
0.0 to 1.0
ADVICE: The specified maximum wall thickness 2.0 is acceptable for selected
material acetal and process injection-moulding (maximum thickness for process
is 3.6 ) (GENR2)
ADVICE: The specified minimum wall thickness 1.5 is acceptable for selected
material acetal and process injection-moulding. (minimum thickness for
material is 0.8)
ADVICE: The specificed draft angle  0.0  degrees is less than the minimum
draft angle for a part manufactured using injection-moulding you are
recommended to reduce the increase the draft angle to at least 0.5 degrees
(GENR5)
ADVICE: Moulded parts should be designed with uniform wall thickness (GENR1)
INFORMATION: The maximum main wall thickness on the part is 2.0 and the
minimum main wall thickness is 2.0         ADVICE: The variation in the main-
```

```
wall thickness on the part is 0.0 percent which is acceptable for a part where
appearance is not a consideration (less than 0.5)
ADVICE: The design should not have abrupt section changes. Where a section
change is required a gradual taper of 3.0 must be applied (GENR6)
ADVICE: All Corners should be generously radiussed (GENR7)
INFORMATION: Rib 67 has thickness  1.5 mm which is 75.0 percent of the main
wall thickness (2.0 mm )
ADVICE: For process injection-moulding the recommended rib thickness is 60.0
percent of main wall thickness
Rib 67 is too thick and should be reduced to 1.2
INFORMATION: Rib 63 has thickness  1.5 mm which is 75.0 percent of the main
wall thickness (2.0 mm )
ADVICE: For process injection-moulding the recommended rib thickness is 60.0
percent of main wall thickness
Rib 63 is too thick and should be reduced to 1.2
INFORMATION: Rib 59 has thickness  1.5 mm which is 75.0 percent of the main
wall thickness (2.0 mm )
ADVICE: For process injection-moulding the recommended rib thickness is 60.0
percent of main wall thickness
Rib 59 is too thick and should be reduced to 1.2
INFORMATION: Rib 55 has thickness  1.5 mm which is 75.0 percent of the main
wall thickness (2.0 mm )
ADVICE: For process injection-moulding the recommended rib thickness is 60.0
percent of main wall thickness
Rib 55 is too thick and should be reduced to 1.2
INFORMATION: Rib 45 has thickness  2.0 mm which is 100.0 percent of the main
wall thickness (2.0 mm )
ADVICE: For process injection-moulding the recommended rib thickness is 60.0
percent of main wall thickness
Rib 45 is too thick and should be reduced to 1.2
INFORMATION: Rib 44 has thickness  2.0 mm which is 100.0 percent of the main
wall thickness (2.0 mm )
ADVICE: For process injection-moulding the recommended rib thickness is 60.0
percent of main wall thickness
Rib 44 is too thick and should be reduced to 1.2
INFORMATION: Rib 43 has thickness  2.0 mm which is 100.0 percent of the main
wall thickness (2.0 mm )
ADVICE: For process injection-moulding the recommended rib thickness is 60.0
percent of main wall thickness
INFORMATION: Rib 42 has thickness  2.0 mm which is 100.0 percent of the main
wall thickness (2.0 mm )
ADVICE: For process injection-moulding the recommended rib thickness is 60.0
percent of main wall thickness
Rib 42 is too thick and should be reduced to 1.2
INFORMATION: Rib 36 has thickness  2.0 mm which is 100.0 percent of the main
wall thickness (2.0 mm )
ADVICE: For process injection-moulding the recommended rib thickness is 60.0
percent of main wall thickness
Rib 36 is too thick and should be reduced to 1.2
INFORMATION: Rib 33 has thickness  1.5 mm which is 75.0 percent of the main
wall thickness (2.0 mm )
ADVICE: For process injection-moulding the recommended rib thickness is 60.0
percent of main wall thickness
Rib 33 is too thick and should be reduced to 1.2
INFORMATION: Rib 29 has thickness  1.5 mm which is 75.0 percent of the main
wall thickness (2.0 mm )
ADVICE: For process injection-moulding the recommended rib thickness is 60.0
percent of main wall thickness
Rib 29 is too thick and should be reduced to 1.2
INFORMATION: Rib 17 has thickness  1.5 mm which is 75.0 percent of the main
wall thickness (2.0 mm )
ADVICE: For process injection-moulding the recommended rib thickness is 60.0
percent of main wall thickness
Rib 17 is too thick and should be reduced to 1.2
INFORMATION: Rib 15 has thickness  2.0 mm which is 100.0 percent of the main
wall thickness (2.0 mm )
ADVICE: For process injection-moulding the recommended rib thickness is 60.0
percent of main wall thickness
Rib 15 is too thick and should be reduced to 1.2
INFORMATION: Rib 9 has thickness  1.5 mm which is 75.0 percent of the main
wall thickness (2.0 mm )
ADVICE: For process injection-moulding the recommended rib thickness is 60.0
percent of main wall thickness
```

```
Rib 9 is too thick and should be reduced to 1.2
```