Chapter 7
Concurrent Composite Structural Optimization

This chapter demonstrates a new methodology for integrated optimization of composite structures. A Knowledge-Based library of modeling tools for composite rib was developed. The library includes algorithms for creating a number of features that appear on real composite ribs and a novel method for capturing the complex logic involved in decision making during manufacturing process selection. The knowledge related to structural optimization was captured in the form of specialized FEA modeling entities, and a link to the ANSYS (ANSYS, Inc.) package was demonstrated. Also incorporated was an algorithm similar to that used in Process Link (Section 5.4) to build a process model from high level decisions regarding manufacturing methods.

The rib model of this demonstration was not intended to be highly accurate. Rather, the objective was to demonstrate how knowledge relating to all the disciplines involved in composite rib design optimization could be captured. Also, no use of optimization algorithms was made since a more fundamental form of structural optimization is presented whereby the configuration, topology, sizing and shape are all considered to be design variables. In general, the manipulation of these diverse variables is not possible with numerical optimization tools. Thus, the focus was on demonstrating a framework in which a wide variety of optimization methods could be applied. The reader should refer to the first three chapters of this thesis for examples of the optimization methods that could be integrated into this system.

7.1 Methodology

A prototype software system for a new generation of Knowledge-Based concurrent engineering design was developed as part of this research. The exact specifications are not discussed here for intellectual property reasons however the important attributes of the system are discussed in the following sections.

7.1.1 System Architecture

Figure 7.1 is a diagram showing the principle modules and data entities that make up the prototype system. At the center of the diagram lies the interactive modeling environment for processing information relating to a number of external models and data sets. Inputs to the environment are CAD data, Manufacturing Data and the Knowledge Library that is created during prior modeling sessions. The user interacts with the data directly through the modeling environment's Graphical User Interface (GUI) and Geometric Modeler, or indirectly through 3'rd Party Software. The outputs from the system are Images, Product Models and Technical Data Summaries.

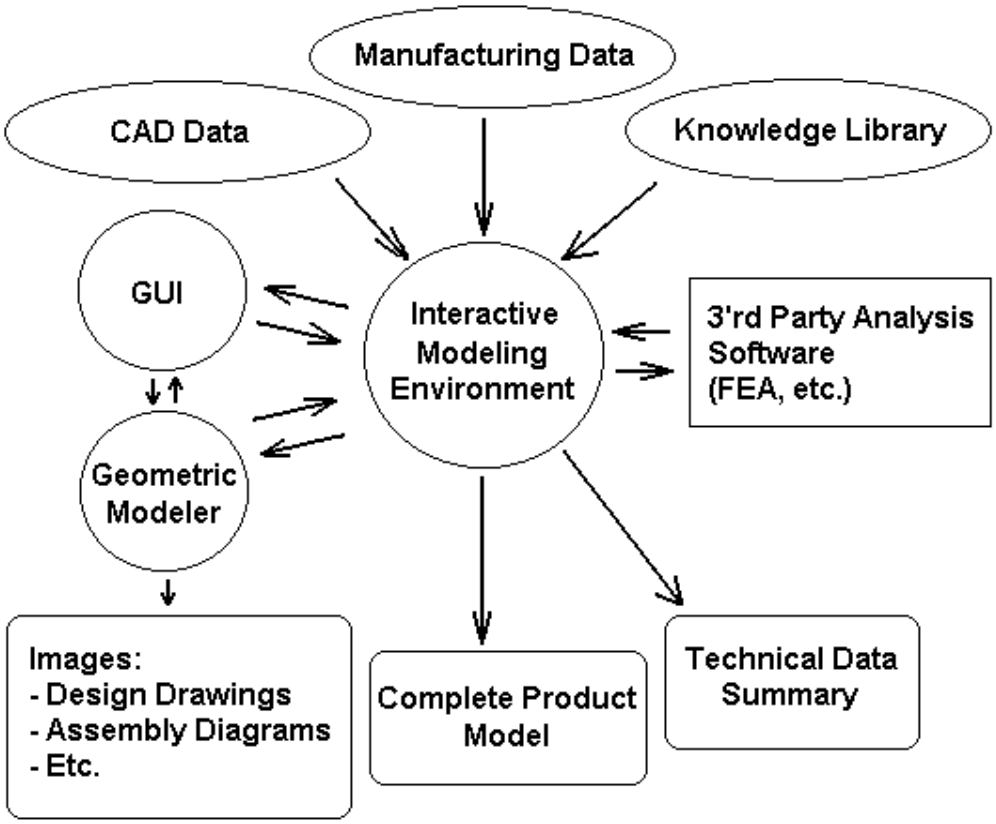

Figure 7.1, Modules and data sets for the prototype Knowledge-Based system

Some estimates were made to give an idea of the complexity and size of the C++ software that implements this prototype system. The system contains over 150 classes, the average class containing about 10 functions, and the average function

containing about 20 lines of code. This give a rough estimate of 30,000 lines of code that were created, tested and debugged in the 8 month period during which it was developed.

Considering the effort involved in making this rudimentary implementation of the concepts described above, it is understandable why the large CAE software vendors did not create such a system decades ago. Commercial quality software requires far higher robustness than research code. It is perhaps fair to say that such companies have concentrated on leveraging the benefits of the software they already own through continuous small improvements, even when the basic methodologies of that software are no more advanced than those of 20 years ago. This lag of concept implementation is thought to be a main cause of the dichotomy that was proposed in Section 6.1.3.

7.1.2 Use of EXPRESS-2

The most fundamental aspect of the system is that it includes, at its core, the EXPRESS-2 language (Spilby & Sanderson, 1999) from the STEP project of the ISO. This is useful in many ways. Firstly, the interfacing issues that are mentioned in Section 6.1 are essentially avoided since the STEP Application Protocols (written in EXPRESS) are rapidly emerging as the standard for CAE data exchange. Secondly, EXPRESS-2 is an Object-Oriented language and hence it can naturally be specialized to form part of a customizable Knowledge-Based system (according to Section 6.3.3). Finally, the EXPRESS-2 language uses some useful technologies[1] developed for the JAVA language. Hence, it is relatively simple to program with (compared to some other Object-Oriented languages such as C++. The development of an easy to program, customizable system was seen as crucial for designers that want to utilize the benefits of Knowledge-Based techniques but cannot apply the hard-wired features and knowledge entities that are built in to existing commercial systems.

[1] These technologies include automatic garbage collection and reference counting. A discussion of the merits of these technologies is beyond the scope of this thesis.

Before developing the proposed system, two existing software systems for information modeling with the EXPRESS language were reviewed. Libes (1993) developed the NIST EXPRESS toolkit – a software test bed in which the evolving STEP Application Protocols could be benchmarked. It included an EXPRESS compiler, a dictionary of entity types and a database for stored instances of the types. The software for the system was written in the C language. Sauder & Morris (1995) developed a C++ software toolkit to provide building blocks for developing STEP software applications. This linked the NIST EXPRESS toolkit with other modules that are useful for generating EXPRESS schema files[2]. In the software developed during this research, source code from these systems was not utilized. However, some general aspects of their designs were used as a starting point.

In a similar manner to Libes (1993), the system developed for this research used an information database to store instances of the entity types defined by the EXPRESS-2 code. One key function of this database is to support input and output of STEP compliant P21 files. This allows the system to share data with any other STEP compliant CAE system – an important issue in modern design practice where different organisations own and operate different software.

7.1.3 The In-Built Geometric Modeler

Another key aspect of the system is its geometric modeler. The usefulness of this was discussed in Section 6.4.1. Many geometry display functions from the Process Link program (discussed in Section 5.4) were reused in this system. Due to the complexity (and hence monetary cost) of geometric modelers, only rudimentary geometric modeling operations were included. The in-built geometric entities are detailed in Appendix D and were limited to linear geometry (i.e. lines, and planar faces). The modeler cannot represent solids or curves directly, however, storing, manipulating and displaying the raw data for almost any sort of geometry including

[2] EXPRESS schema files are the collection of all entity types, functions rules etc. that define a particular information modeling solution. The STEP Application Protocols take the form of EXPRESS schema files.

curved surfaces and solids is possible by writing the necessary EXPRESS-2 code to customize the system.

The EXPRESS-2 code that customizes the system is slower to execute than the precompiled code used for its creation. Hence, calculation intensive operations are best done with in-built functions. For the rib demonstration the geometric modeling operations were the most complex so a set of fundamental geometric functions was built into the system. These functions were low-level operations for manipulating *poly_loop* entities (see Appendix D) and would be internal to most CAD systems. To allow maximum flexibility, these functions were readily accessible with function calls from within the EXPRESS-2 code

7.1.4 Knowledge Library

From discussions with the CRC-ACS staff, a knowledge library for composite ribs was developed. This collection of knowledge entities was defined to encompass all the various types of data used in concurrent design and optimization of composite wing ribs. The entities used in the models are organized in a hierarchical manner. This intuitive approach minimizes complexity when using the system since the designer can concentrate on one particular aspect of the design by focussing attention on a set of closely related branches. The full EXPRESS-2 listing for the user-defined entities that were developed for this research is provided in Appendix E.

7.1.5 Functions

A collection of functions was made during the modeling activities. The modularity of the system meant that each of these functions could be quickly tested and debugged independently during their development - an important factor in the usability of a flexible knowledge capturing system. EXPRESS-2 functions are quite powerful with respects to the modeling actions they can enact. Like functions or procedures in any true programming language, they are able to control program flow (with ‘IF’ and ‘REPEAT’ statements) and call nested functions. The development of software to

compile and execute functions that are written in the draft EXPRESS-2 language was a significant and time consuming activity that was at the core of this research effort.

7.2 Demonstration: Customized Geometric Features

7.2.1 The Rib Entity

Figure 7.2 shows the rib entity type and some of its members. This was the highest level entity type and it references all other entity types either directly or indirectly.

EXPRESS-2 models, being Object-Oriented, can start from any arbitrary level of abstraction. Through member sub-branches the design's various attributes are represented, right down to the 'primitive' entities that make up the most fundamental data. The primitive entities are of one of the following types: INTEGER, REAL, STRING, BOOLEAN, LOGICAL or BAG. The BAG type is particularly useful since bag entities can be used in many situations to represent design information in the form of collections. Examples include the set of points that make up a boundary, the stack of plys making up a laminate and the set of mouse holes for a particular rib design. The 'UNIQUE' modifier (see Figure 7.2) forces the bag's contents to be unique. When this modifier is employed, the bag behaves like a set as per set theory in mathematics. Various useful set operations such as union ($\cup$) and intersection ($\cap$) can be performed on such bags to capture knowledge relating to the product's design.

The first four member branches, 'chord', 'h_on_c', 'profile' and 'loads', relate to fixed information that is set out in the rib's design requirements. The next two members 'features' and 'laminate' relate to flexible data whose values are at the discretion of the designer. Data of this type can be made the subject of some form of optimization (numerical or otherwise). Typical optimization variables for composite ribs are the geometric attributes of the features and the properties of the plies used in the laminate.

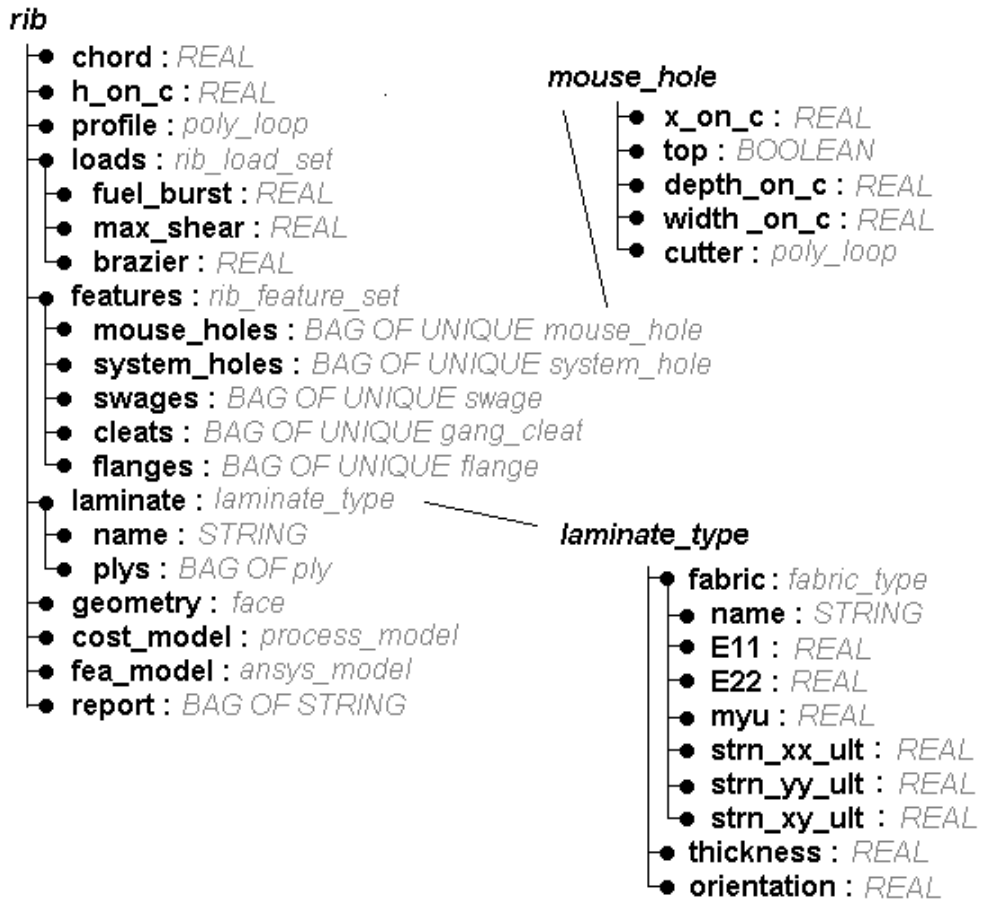

Figure 7.2, Rib entity members (members are of the types indicated in gray)

The ‘features’ member references a number of bag type entities that are used to define the features of the rib. The ‘mouse_hole’ feature represents a notch that is cut in the rib’s perimeter to allow carry through stiffeners to be attached to the inside of the wing’s upper and lower skins. The ‘system_hole’ features are included to allow piping, cabling, control rods or other aircraft systems to connect on both sides of the rib.

The ‘swage’ features are raised regions on the rib that offer some buckling resistance. The CRC-ACS has recently been active in developing manufacturing technologies for swaged composite ribs. The CRC-ACS has also been investigating ‘gang_cleat’ features - ‘L’ shaped components used to attach the rib to the wing with bolts or rivets. An alternative option for forming the rib-to-wing joint is to use

flanges. The ‘flange’ features are used to model 90^0 bends along the rib’s edges used as a mating surface to the inside of the wing.

The ‘geometry’ and ‘report’ members represent design information that is purely a function of the above-mentioned entities. The designer does not choose this data. Rather, algorithms that utilize already-known data determine the properties of these entities. The already-known data is fixed by decisions made earlier in the design process. The reliable and automatic evaluation of this type of data is highly desirable since it is time-consuming for the engineering company yet it does not directly contribute to more optimal designs.

The ’report’ is a simple data structure that contains textual information about the design. Typically, each string in the report bag represents one idea, decision or other piece of knowledge that may be useful for future design tasks. If desired, it can be output as an ASCII text file for subsequent editing with word processing software. Hence, it may be considered the first draft of a technical report on the rib’s design whereby the basic information is given in textual form but still requires conversion to a readable form.

7.2.2 Building the Rib: Initial Stages

This section describes the sequence of steps that were taken to build the composite rib model within the prototype Knowledge-Based system. To begin with, a rib profile is loaded into the system from an external CAD program or by merely specifying the points around the perimeter. The user specifies the exact features to apply to the rib and these are built one-by-one until the full rib geometry is realized.

Figure 7.3 shows the rib profile that was used for this example. Nine vertices can be seen around the profile and straight-line segments join them together (the rudimentary geometric modeler can only model linear geometry). This profile was used to create an area entity that formed the base part for subsequent modeling activities.

Boolean geometric modeling operator functions were developed in EXPRESS-2 to remove the relevant material from this base part. For this demonstration, three system holes features were specified for the model and Figure 7.4 shows the base geometry after system holes were ‘cut’ into it. The locations and radii of these system holes were entirely free and could readily become the subject of design optimization with such a system.

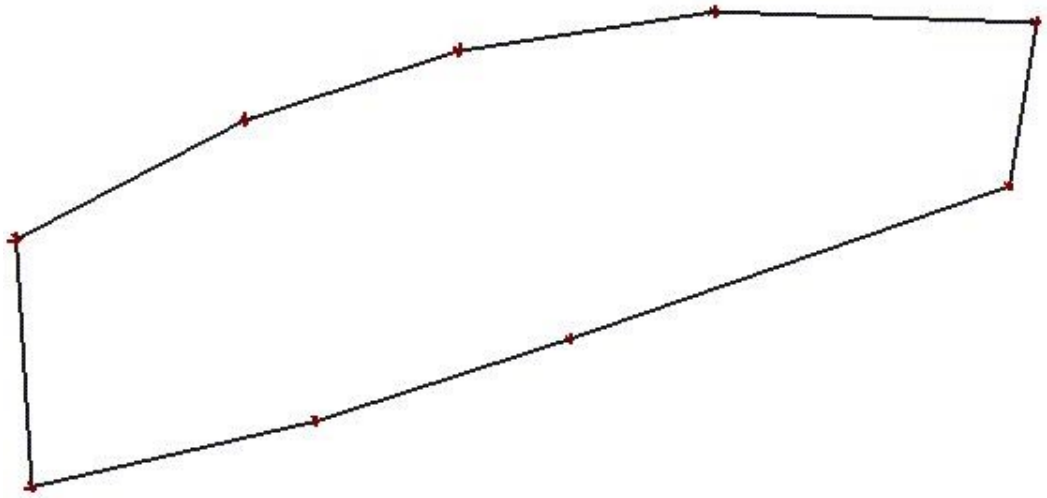

Figure 7.3, Rib profile

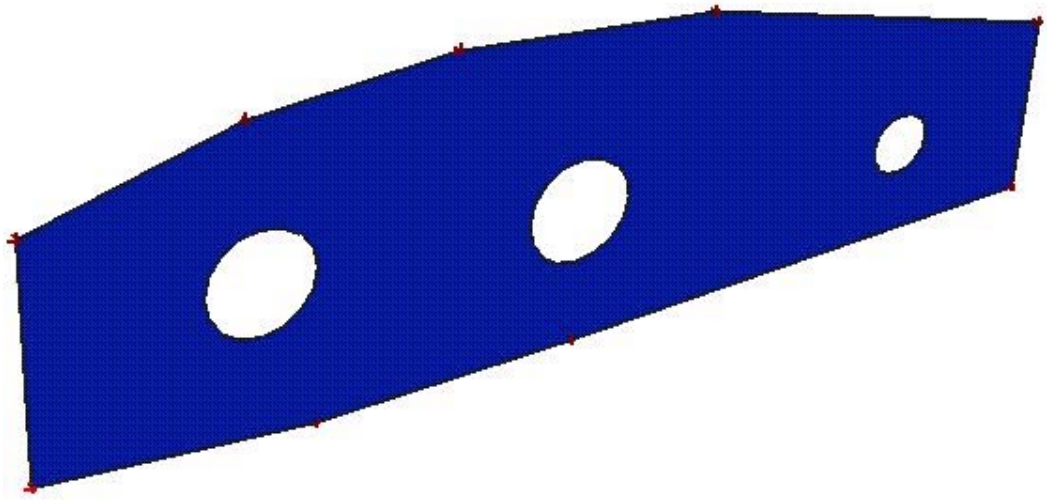

Figure 7.4, Rib with system holes

7.2.3 Building the Rib: Constructing Mouse Holes

After the system hole features were built, mouse hole features were defined and built in a similar way. Figure 7.5 shows the rib after eight mouse hole features were constructed to allow uninterrupted stiffeners on the wing skins – four on the top and four on the bottom.

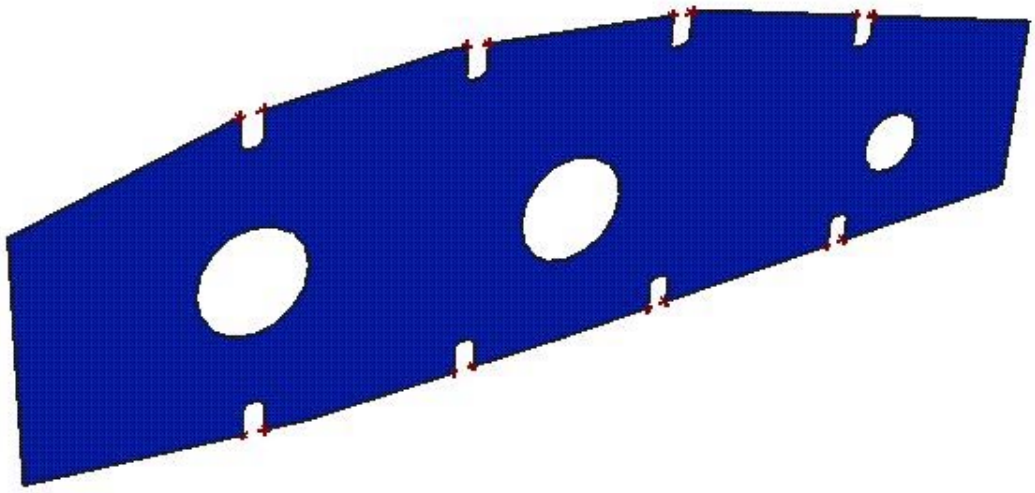

Figure 7.5, Rib with mouse holes

To more clearly explain the functionality of the system, the algorithm for building the mouse hole features is described. This algorithm uses mouse hole location and size parameters to generate a 'poly_loop' instance (see Appendix D) that is used to cut a hole in its parent rib. The EXPRESS-2 code for this function can be seen in Appendix F, and Figure 7.6 shows a flowchart for the algorithm. Initially, the non-dimensionalized mouse hole parameters ('x_on_c', 'width_on_c' and 'depth_on_c') are multiplied by the rib's chord length ('c') to determine the location and dimensions of the mouse hole cut. Then geometric modeling operations are used to get the two points on the rib's profile at the designated x-location of the mouse hole's center point. This gives two new points – one on the top of the profile and one on the bottom. The correct point is then chosen according to the 'top' property of the mouse hole being built. Then the 'cutter' poly_loop is created at the designated x-location and with the appropriate dimensions. Finally, the new poly_loop is used to cut a mouse hole shaped piece out of the rib's geometric model.

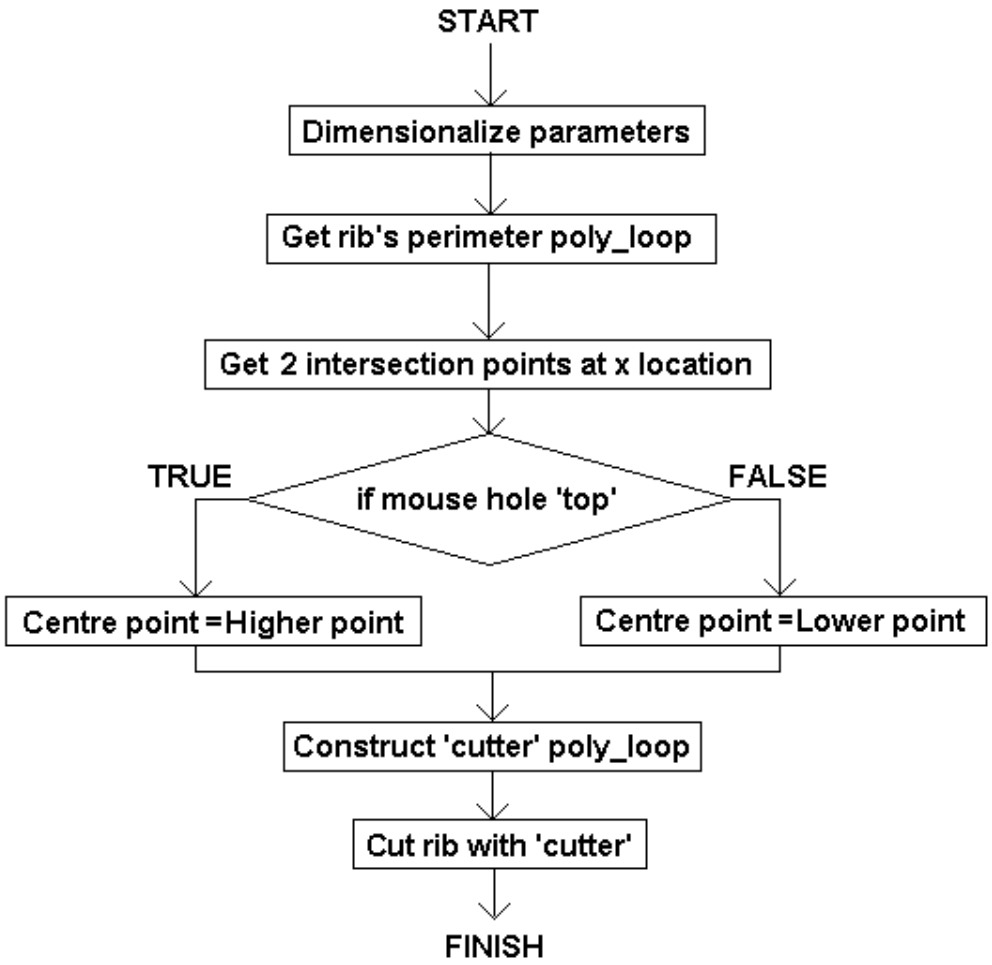

Figure 7.6, Flowchart for the mouse hole building algorithm

Algorithms were also developed for the other features making up the model – system holes, cleats and swages. These algorithms are not discussed here since they utilize similar geometric functionality and the objective of this chapter is merely to show that geometric modeling knowledge can be captured by the system.

7.2.4 Building the Rib: Swages and Gang Cleats

The final features that were built into the rib were the most complex since they involved out-of-plane entities. Firstly, two dissimilar swages were defined to lie between the three system holes and they were constructed using a customised algorithm that removes rectangular pieces of the base part and creates the out-of-plan areas that give the swages their extra stiffness. Figure 7.7 shows the rib after the swages were constructed.

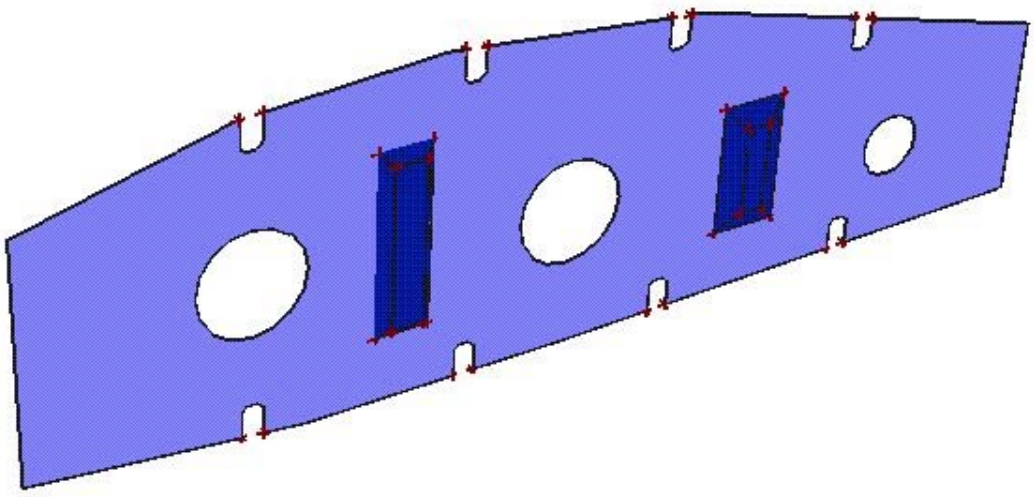

Figure 7.7, Rib with swages

The final stage was the building of gang cleats that connect the rib to the interior of the wing box. The optimal configuration of gang cleats presents an active area of investigation by the CRC-ACS as there are a variety of trade-offs associated with their design. Designs that use two or more gang cleats for a given joint (i.e. ‘rib to top skin’ or ‘rib to bottom skin’) are said to be fail-safe since there is more than one load path. However, manufacturing concerns favour designs with minimum part counts since such designs normally have lower manufacting costs. This trade-off was not addressed in this work, however, the capability of the prototype system to account for it is demonstrated by Figure 7.8. Here, a single gang cleat was specified and built for the connection to the top skin whereas two separate gang cleats were used for the bottom.

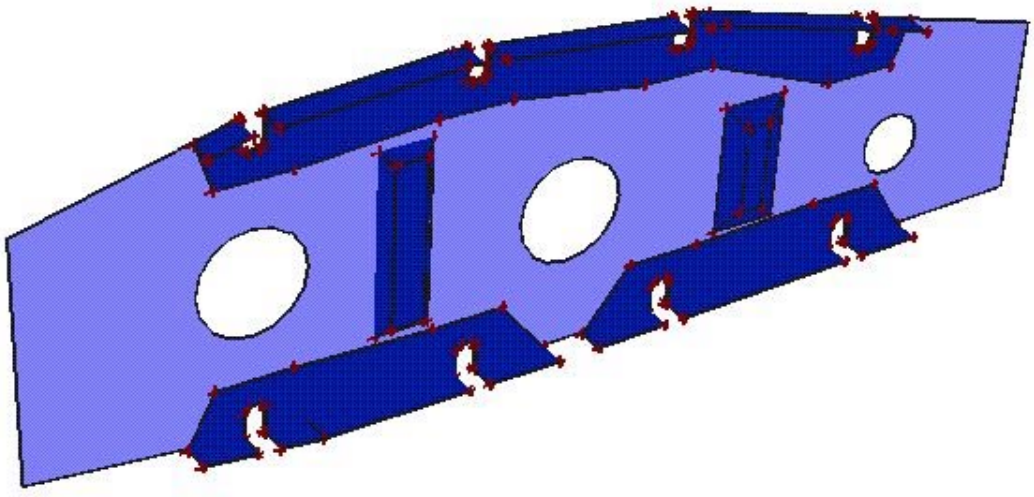

Figure 7.8, Rib with gang cleats

The algorithm that built the gang cleats accounted for the locations and shapes of existing mouse holes since the outer profile for the gang cleats must match that of the rib itself.

7.3 Knowledge-Based Process Modeling

7.3.1 The Process Model Entity

The ‘cost_model’ member of the ‘rib’ entity type (see in Figure 7.2) is of the type named ‘process_model’. The sub-members of this entity type are shown in Figure 7.9. This contains a hierarchical collection of entities providing an interface between the ribs geometric data and the processes defined in the PCAD process database discussed in Section 5.1.

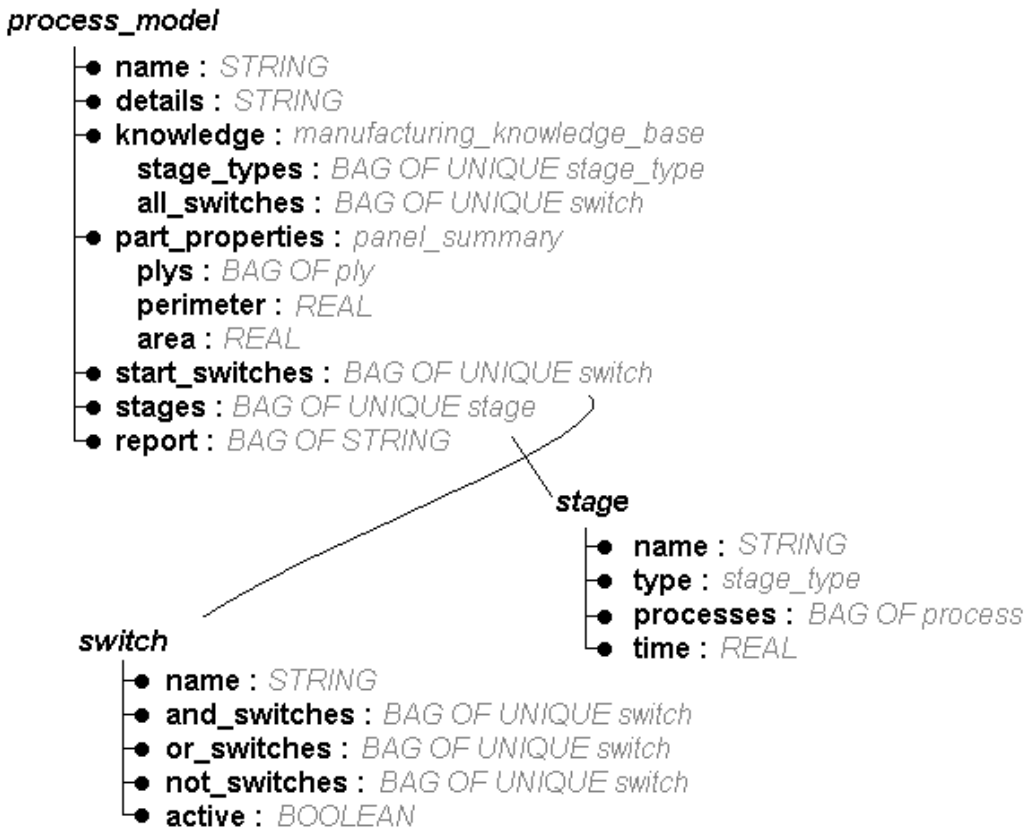

Figure 7.9, The 'process_model' entity type and its members

There is also a 'report' in the process model that was appended to the rib's report after the process models were successfully built.

The 'switch' entity represents knowledge in the form of YES/NO manufacturing decisions. A collection of switches was used to capture the decision graph shown in Figure 7.10. The decision graph is a novel method that was developed to capture knowledge in the form of complex interactions between decisions. Each switch has a state indicator called 'active' that is linked to a variety of other switches with logical relations. Solving the decision graph involves assigning all switches in the knowledge base to either TRUE or FALSE values (initially they were all set to the logical UNKNWON value). Section 7.3.2 discusses the algorithm that was used to solve the decision graph for arbitrary manufacturing scenarios.

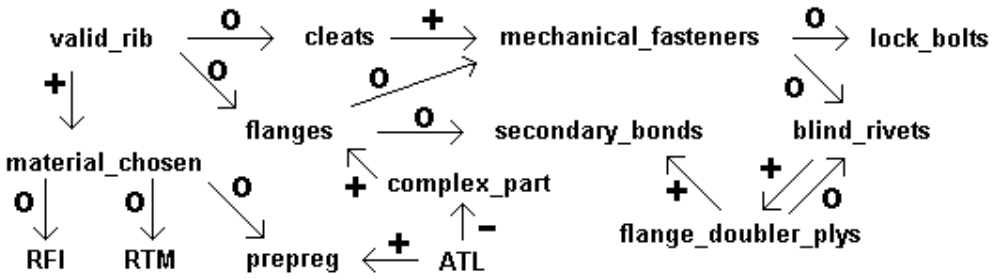

Figure 7.10, Decision graph used in composite panel design
('O', '+' and '-' represent logical OR, AND and NOT respectively)

As an example of the decision graph's purpose, consdier the switch named 'mechanical_fasteners'. This switch has connections to two other swithces – 'lock_bolts' and 'blind_rivets'. The type of the connections is 'O' which indicates as logical OR relationship. Hence, the interpretation of these associations is: "A design mechanical fasteners must have either lock bolts or blind rivets."

7.3.2 Decision Graph Algorithm

This algorithm was used to assist the designer by automatically making sensible assumptions based on previously made decisions. The decision graph of Figure 7.10 formed a complete body of decision switches required for modeling the rib. Some the decisions were trivial (e.g. "Use mechanical fasteners whenever cleats are included"). However, solving the decision graph could also enforce constraints to exist between the decisions. For example, 'valid rib' requires both a material and a method of attachment to the wing to be to be decided. Hence, the 'valid_rib' switch formed a check that all the required decisions were made.

The algorithm starts with a set of start switches that correspond to decisions made either implicitly or explicitly by the designer during the feature creation stage (e.g. fastener types and material types). From these starting points the algorithm attempts to determine the state of every switch in the manufacturing knowledge base's

'all_switches' bag entity (see Figure 7.9). The algorithm loops through trying to propagate all the switches until a terminating event occurs. One possible terminating event occurs if the propagation of the switches fails to reach all switches in the model. In this case the system reports the discrepancy to the user. The user then alters the contents of the process model's start switches bag in an attempt to provide the system with a valid and complete set of switches to start the propagation.

The other possible terminating event occurs when the switches propagate fully. In this case, the algorithm reports to the user the state of all the switches. Figure 7.11 shows the flow chart for the algorithm.

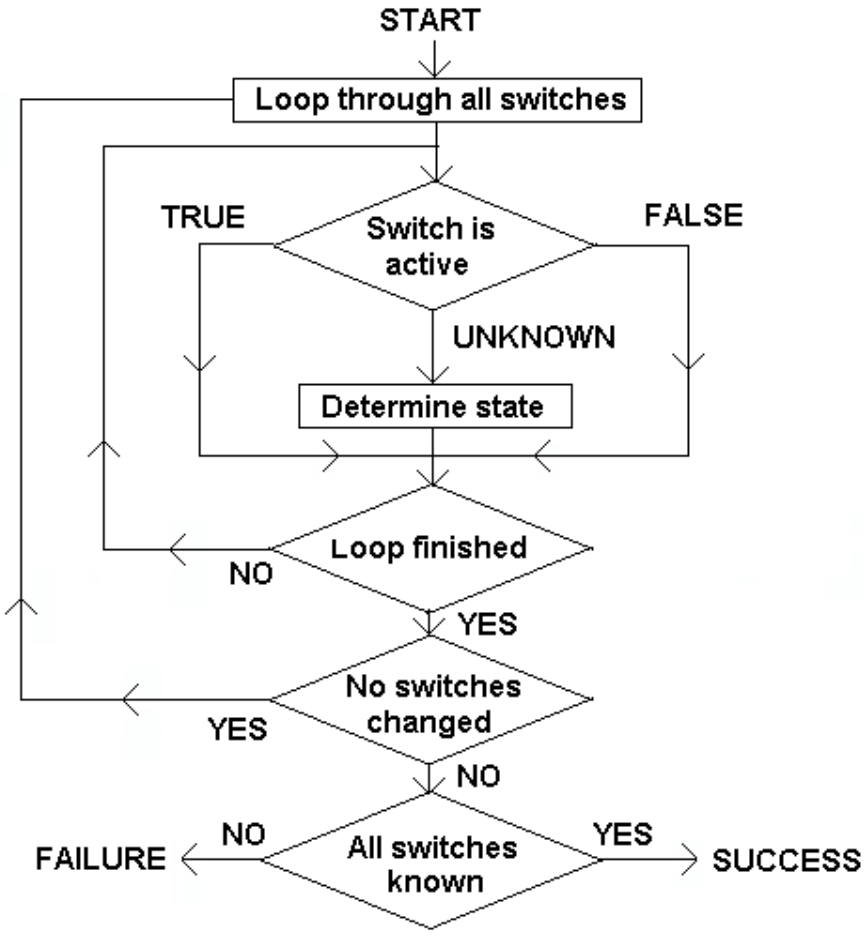

Figure 7.11, Decision Graph Solving Algorithm

7.3.3 Process Selection Algorithm

The purpose of this algorithm was to automatically select the processes required for the manufacturing of the product. Further developments may lead to algorithms that optimize the selection of these processes. However, in this implementation, the process selection took the same form as was used in the Process Link software (Section 5.4). The switches that determine the manufacturing properties of the part (as determined by the Section 7.3.2) are used to select the relevant 'stage' types for the manufacturing sequence (lay-up of plies, autoclave cure cycles, etc.). As shown in Figure 7.9 each of the 'stage' entities has a bag of process steps that correspond to a set of PCAD cost equations. The flow chart for this algorithm is shown in Figure 7.12

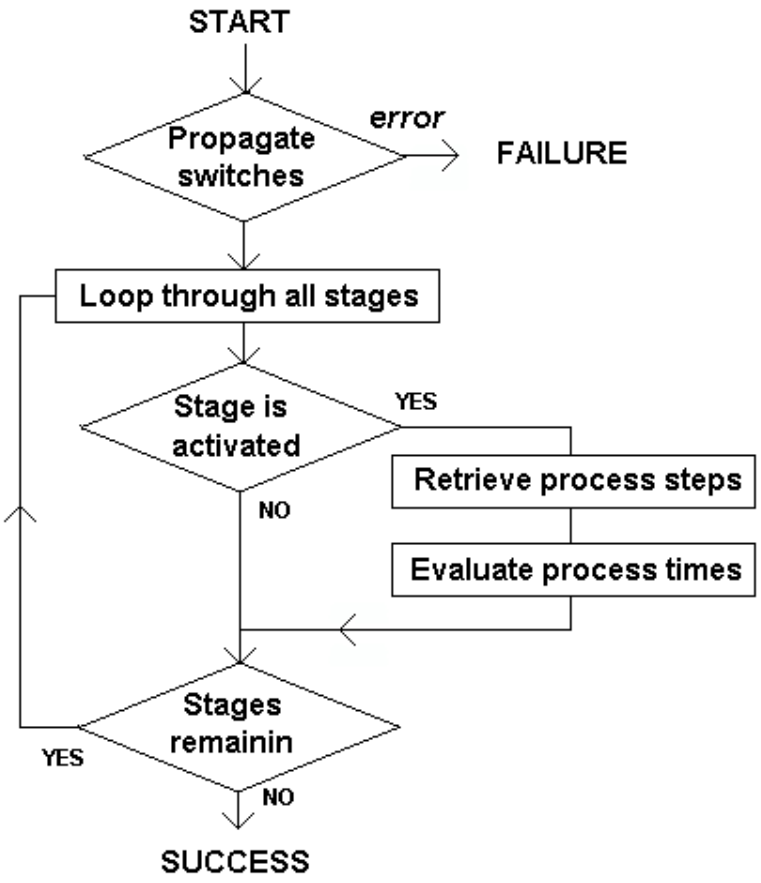

Figure 7.12, Process selection algorithm.

7.3.4 Process Cost Modeling

A cost model for the featured rib model was generated with the following starting switches: ‘cleats’, ‘lock bolts’ and ‘RTM’. These three switches fully determined the state of the decision graph of Figure 7.10. The following table indicates the solved states of all the switches.

Switch Name	**Active**
valid_rib	TRUE
cleats	TRUE
mechanical_fasteners	TRUE
lock_bolts	TRUE
blind_rivets	FALSE
flange_doubler_plys	FALSE
flanges	FALSE
secondary_bonds	FALSE
complex_part	FALSE
ATL	FALSE
material_chosen	TRUE
prepreg	FALSE
RTM	TRUE
RFI	FALSE

Table 7.13, States of the switches in the solved decision graph.

With these switches used, the process selection algorithm automatically determined that the following stage types were required: ‘Prepare Tool’, ‘Prepare RTM Fabric’, ‘Cure in Oven’, ‘Extract Part’ and ‘NDI’. The time taken for each of these stages was broken down into the individual processes. Each of the relevant processes was included in the process model and the process times were evaluated. Simultaneously, the process model report was generated.

Figure 7.14 shows a column chart of this time breakdown. The column graph clearly indicates the RTM Cure stage was most time consuming. Such an observation would suggest to the designer that a change to prepreg material might shorten the manufacturing time. Hence, graphs such as this could be used for rapid feedback to the designer when they are exploring the impact of their manufacturing decisions.

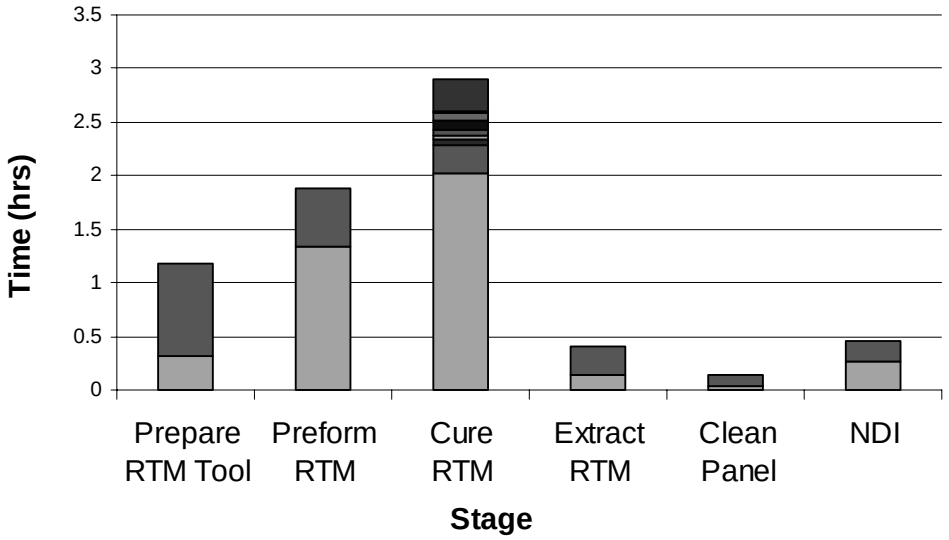

Figure 7.14, Process time breakdown for the rib manufacturing model (only 'stage' data is shown for simplicity)

7.4 Highly Automated Finite Element Analysis

An integrated Knowledge-Based engineering environment requires a strong linkage to Finite Element Analysis (FEA) capabilities for activities such as redesign, structural optimization and automated model building and analysis. Thus, the inclusion of FEA into this modeling demonstration was essential. To this end, an ANSYS FEA model entity was developed to run on the prototype system and this entity is described in the following section. The composite rib's design data was used to create the model and the ANSYS program was used to perform linear static and linear buckling analysis of the structure.

7.4.1 The ANSYS FEA Model Entity

To link to ANSYS, a number of entity types related to finite element data storage were formulated. The 'ansys_model' entity is basically a collection of finite element data entities stored in bags. Figure 7.15 shows the various members of this entity type. The 'ansys_area' and 'point_bc' entity types represent a planar bounded surface (face) and a point boundary condition respectively[3].

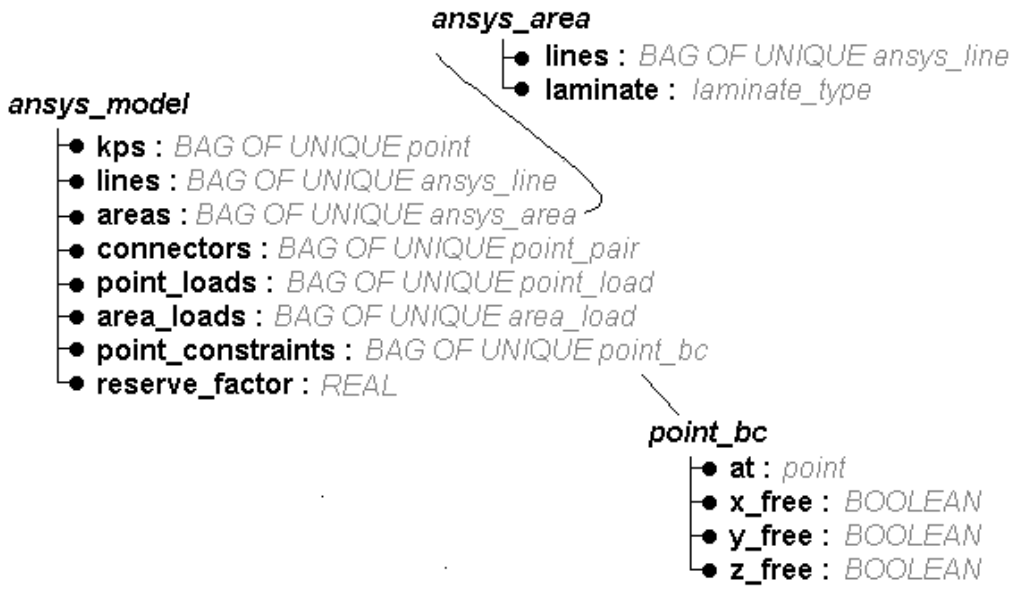

Figure 7.15, The 'ansys_model' entity type and its sub-entities

[3] Further discussion of these entity types is not included since they are largely self-explanatory to people who are experienced with finite element analysis.

7.4.2 Finite Element Analysis

Figure 7.16 shows the ANSYS finite element model that was constructed[4]. This model was taken from the featured rib model shown in Figure 7.7 (the gang cleats were omitted from the FEA model for simplicity). For the purposes of this demonstration it was assumed that the rib was loaded in shear. This is indicative of the linear static loads on a rib during a maximum wing bending load case.

Customized algorithms (not shown here) were developed to instruct ANSYS to perform linear static and linear buckling analysis of the rib. This level of FEA automation presents the possibility for future advancements to the field of structural optimization since the Feature-Based design definitions shown earlier can be included in the optimization formulation.

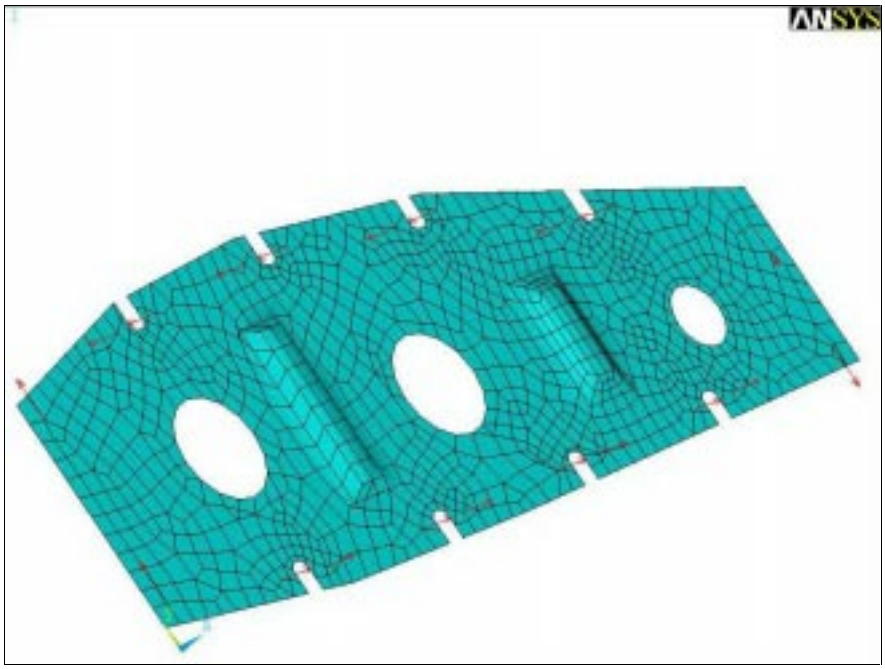

Figure 7.16, ANSYS finite element model of the rib
(arrows near the mouse holes indicate shear loads)

[4] The details of this model are not discussed here since they merely follow standard techniques and do not represent a significant contribution to the field.

To clearly demonstrate the linkage between the prototype system and ANSYS, the buckling response of the design was investigated as this is of particular relevance to the dimensioning of the swages. Figure 7.17 shows the ANSYS analysis model of the rib in a buckled state. Note that presence of the swages has resulted in a shear buckling 'blister' that is predominantly around the middle system hole. Knowledge gained from such FEA could be used when choosing the properties of the swages used for arbitrary rib designs.

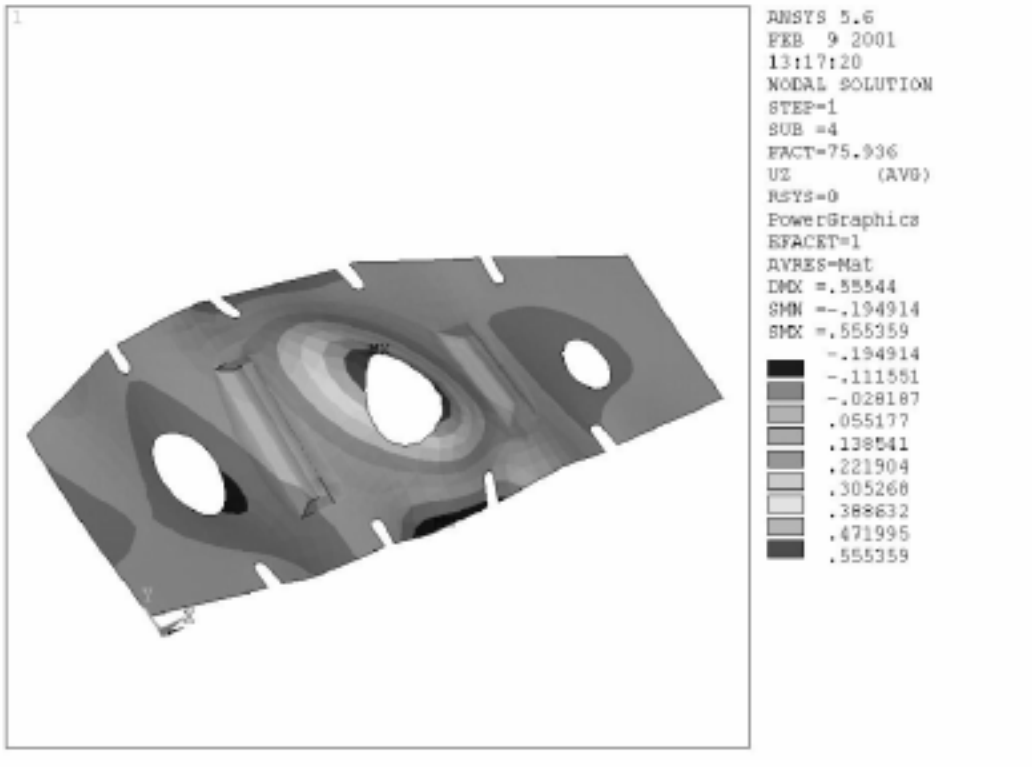

Figure 7.17, ANSYS buckling analysis of the rib.

7.5 Conclusion

This work has clearly demonstrated that the prototype system specified here is able to capture the broad array of knowledge types necessary for advanced concurrent engineering design and optimization of composite structures. The integrated nature of the system presented a significant improvement over existing CAE methods.

Optimization of different types could be included for many of the feature parameters discussed in this chapter. Also, future developments could see optimization of the choice of features selected for the rib's design. For instance, topological optimization could be used for determining swage or system hole locations. Alternatively, sizing optimization could be applied to determine optimal ply stacks or cleat dimensions.

Although no such algorithms were demonstrated in the system, it was clearly shown that any algorithm could be applied in this context – mathematical or non-mathematical, discrete or continuous, optimization or other. As such the methodology presented should be considered more as test-bed or collection of tools for performing structural optimization rather that an optimization solution in itself.

The results presented above represent only the tip of the iceberg as far as the potential for this approach. However, now that the basic functionality of the system has been demonstrated, the wide-ranging concurrent engineering concerns can be incorporated with relatively little effort and little risk of failure. Some of the improvements to the models and the system are suggested below.

7.5.1 Improved Rib Models

The link between decision graphs and feature creation could be improved to allow geometric constraints to be more thoroughly addressed during conceptual design. One example is automatically including cleat features in the rib's design when the user specifies the 'cleats' switch[5]. Another example would be assigning laminate properties to the rib according to material choice switches.

Other methods of predicting manufacturing cost could involve the following factors: geometric tooling issues, factory layout data, uncertainty of successful process completion and staff allocation. The development of data structures and algorithms for these factors would be a time consuming yet valuable activity.

[5] In this work, the start switches were manually set after the features were built.

Libraries of common design information would need to be extensively enhanced before this system could be applied to real design tasks. Most engineering organisations have an extensive body of existing data such as ply stacking sequences, mechanical fastener properties, material properties and geometric constraints for fabrication. If this data could be progressively integrated into Knowledge-Based systems the company's productivity and cost-efficiency would surely increase.

The system was built such that EXPRESS-2 code could be written to call entire FEA analysis cycles repeatedly for tasks such as ply stack optimization, system hole placement or swage geometry optimization. Furthermore, the ANSYS optimizer could also be called from the system or vice versa. Hence, one possible avenue for further research would be to incorporate some of the structural optimization techniques demonstrated in the earlier chapters of this thesis.

7.5.2 Improvements to the Prototype System

Although the functions and entities presented in this chapter were able to capture a large quantity of design knowledge, support for the full ISO version of EXPRESS-2 language would be highly beneficial. In particular, support should be given for inheritance, lists, arrays, type variables and multiple schema files before more substantial feature libraries are developed. The reader is referred to Spliby & Sanderson (1999) for a complete description of these aspects of the EXPRESS-2 language.

As discussed in Section 7.1.3, the geometric entities that could be displayed by the in-built geometric modeler were limited to linear, non-solid geometry. Better geometric modeling is essential for real-world design tasks. Now that the system has been shown to be practical for simple geometry, the option of modifying the system to include advanced nonlinear geometry has become more attractive.

References

ANSYS, Inc., *ANSYS Structural Analysis Guide 4th Edition*, ANSYS Inc. Canonsburg, PA, USA, 1998.

Libes, D., 'The NIST EXPRESS Toolkit – Design and Implementation', Report, Factory Automation Systems Division, NIST, Maryland, 1993.

Sauder, D. A., Morris, K. C., 'Design of a C++ Library for Implementing EXPRESS: The NIST Class Library', *EXPRESS User Group: October*, 1995.

Spilby P., Sanderson D., "EXPRESS Language Reference Manual", ISO TC184/SC4/WG11 N61, 1999.