CRANFIELD UNIVERSITY

S REHMAN

Knowledge-Based Cost Modelling for Innovative Design

CRANFIELD COLLEGE OF AERONAUTICS

2000

PhD

for

my parents,

my husband, Reza

and

my daughter, Sabrina.

ABSTRACT

The contribution to new knowledge from this research is a novel method for modelling production costs throughout the design phase of a product's lifecycle, from conceptual to detail design.

The provision of cost data throughout the design phase allows management to make more accurate bid estimates and encourages designers to design to cost, leading to a reduction in the amount of design rework and product's time to market. The cost modelling strategy adopted incorporates the use of knowledge-based and case-based approaches.

Cost estimation is automated by linking design knowledge, required for predicting design features from incomplete design descriptions, to production knowledge. The link between the different paradigms is achieved through the blackboard framework of problem solving which incorporates both case-based and rule-based reasoning.

The method described is aimed at innovative design activities in which original designs are produced which are similar to some extent to past design solutions. The method is validated through a prototyping approach. Tests conducted on the prototype confirm that the designed method models costs sufficiently accurately within the range of its own knowledge base. It can therefore be inferred that the designed cost modelling methodology sets out a feasible approach to cost estimation throughout the design phase.

Acknowledgements

I would like to thank CarnaudMetalBox Engineering based in Shipley, Yorkshire for their support in this research study. In particular, I would like to thank William Jowitt, David Lupton and David Allen for lending their expertise to this research work.

I would also like to thank my supervisors Dr. M D Guenov, Prof. A J Morris and Dr. J Saggu for their guidance in this research study.

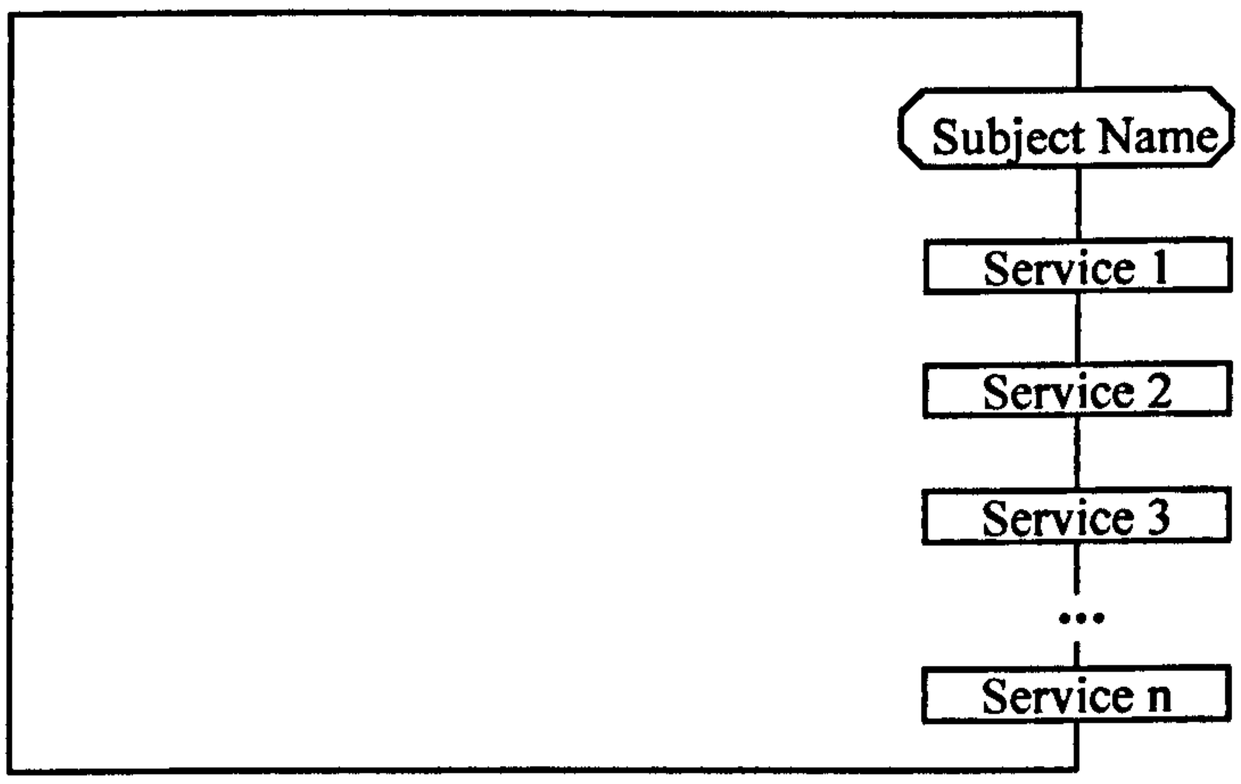
CONTENTS

List of Figures

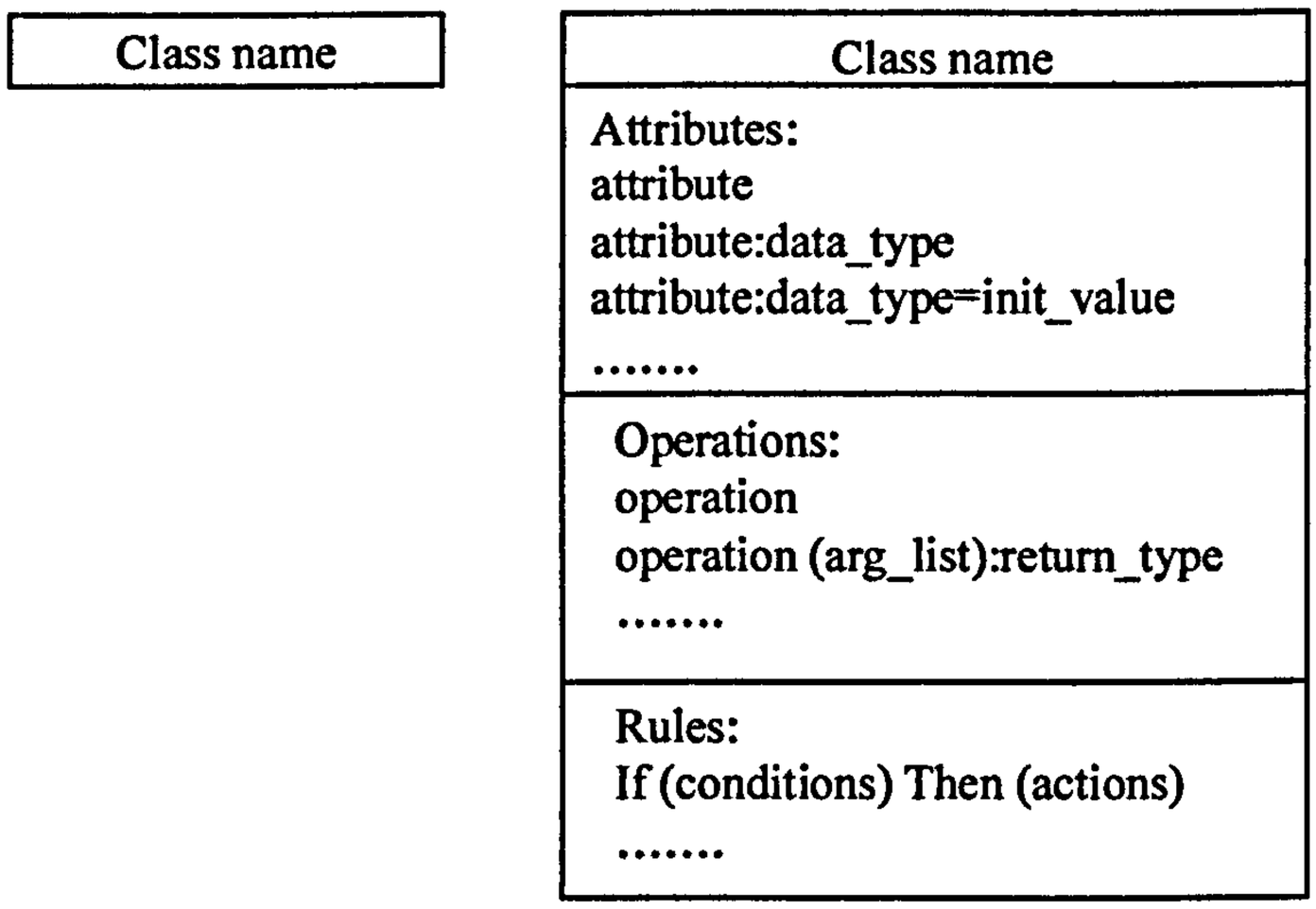
List of Data Tables

Extended OMT Object Model Notation

Subject:



Class:

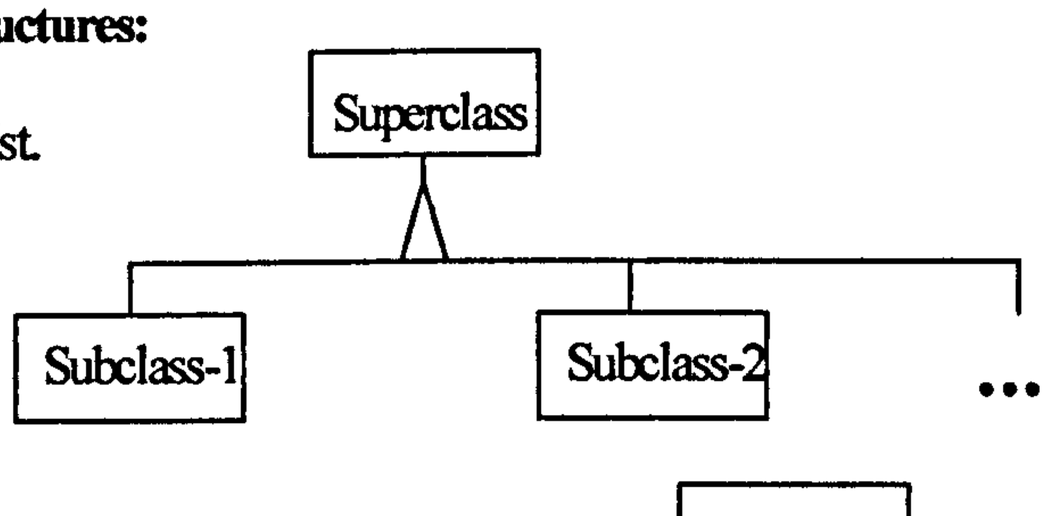


Instances:

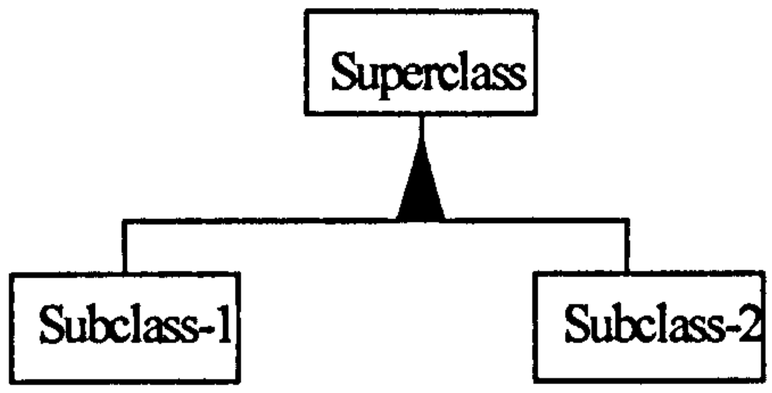
Instantiation relationship:

Classification (is-a) Structures:

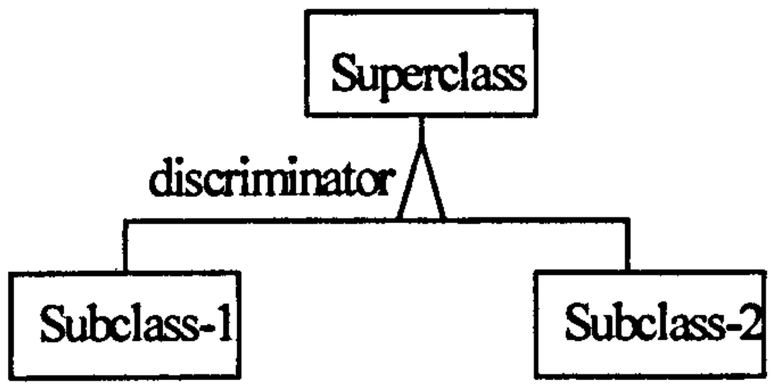
(a) More Subclasses exist.



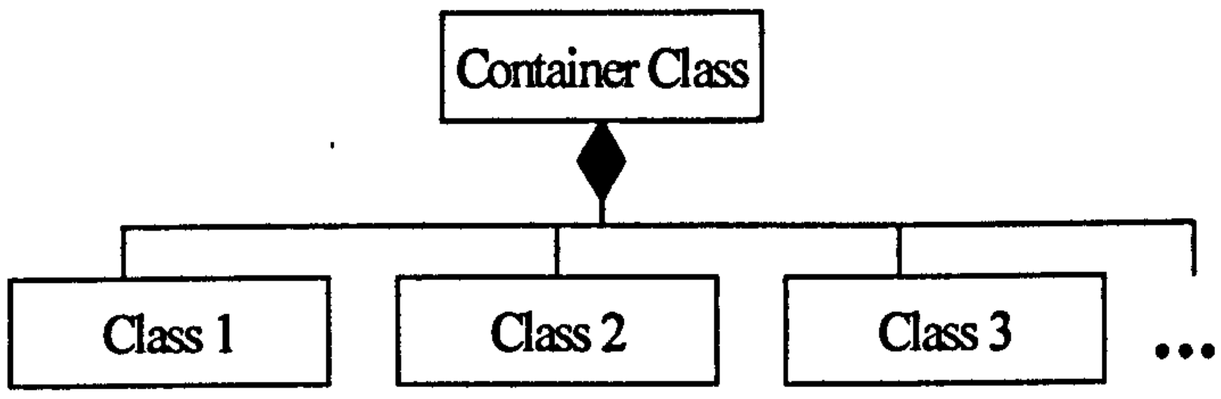
(b) Subclasses have disjoint membership



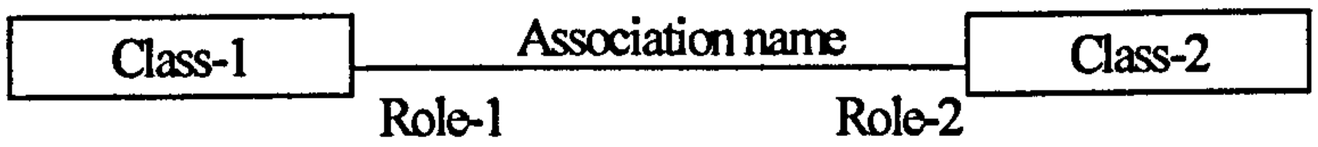
(c) Discriminator is an attribute whose value differentiates between subclasses.



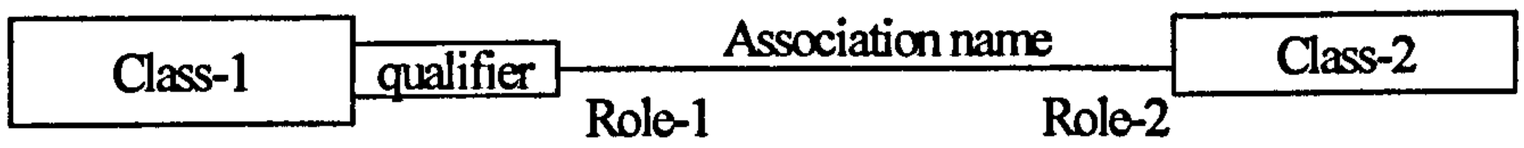
Composition (has-a) Structure:



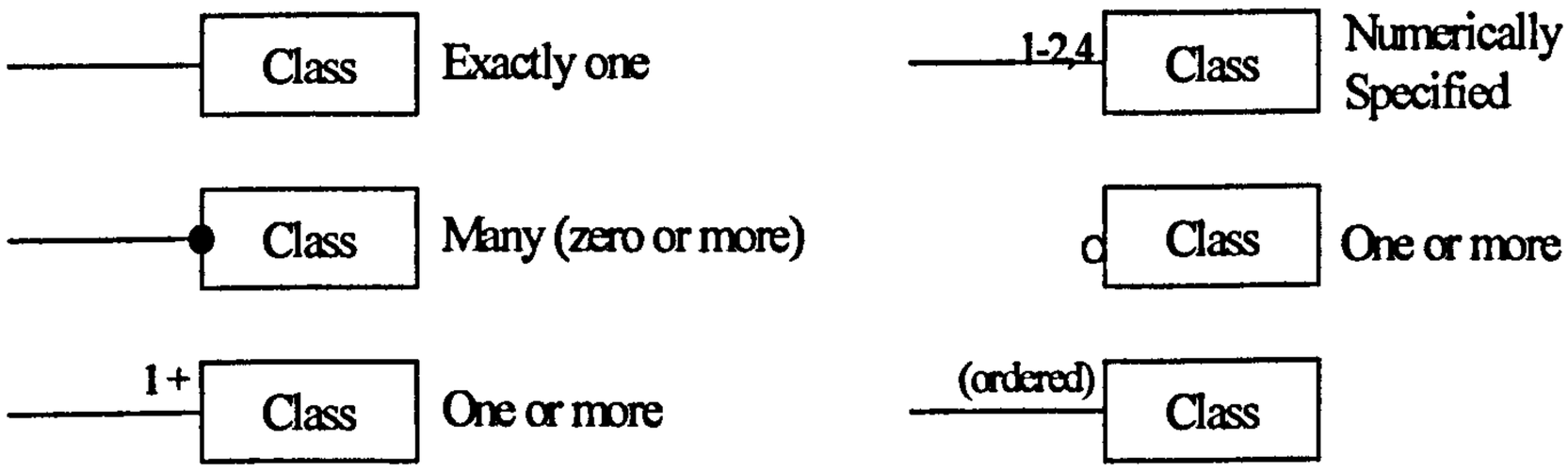
Association:



Qualified Association:



Multiplicity of associations:



1 Introduction

1.1 The need for research

In today's global market place, competitive edge is not only required to gain greater market share but necessary to ensure survival. Among the main factors that influence competitiveness are product costs and time-to-market. Studies have shown that the greatest potential for cost reduction is at the early design phase; where as much as 80% of the cost of a product is decided (*Andreason et al.(1983a), Corbett(1986), Whitney(1988) and Vogt(1988)*). The potential for cost reduction decreases rapidly as the product development process progresses. However, the design phase itself accounts for a relatively small proportion of the total product development cost. Therefore, greater investment in design is a reasonable and necessary step if a company is to remain successful.

Concurrent engineering, in which a team-centred, process-oriented approach to product development is taken, is widely regarded as the vehicle for dramatically improving the design process. The current state of the design process is one of isolation within a vertical, functional framework (Figure 1). This isolated nature of design results in the majority of design changes being made at the trailing end of detail design and at the beginning of production, which leads to costly design rework and a longer time-to-market. The desired state is a team-centred one in which communication is key (Figure 2). In the ideal design process, the majority of changes would take place at the early design phase, where they have the minimum effect on product development costs. The timely provision of product information is essential for allowing the right design choices to be made throughout the design process.

The design process is recognised as a core business process; successfully reengineering it should result in dramatic improvements in a company's performance. Such a reengineering effort however entails high risks.

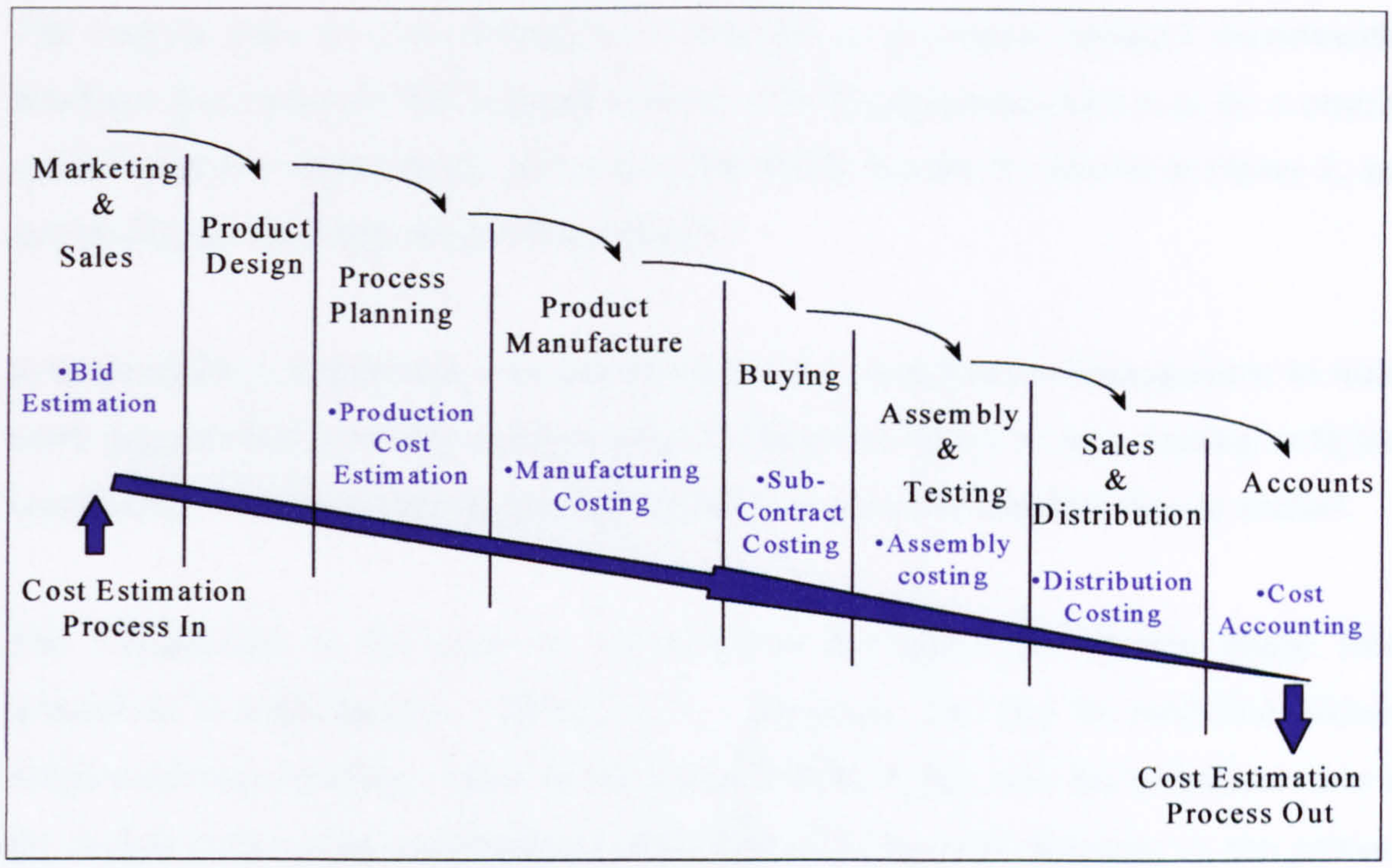


Figure 1. Traditional vs. Process-Oriented Product Development

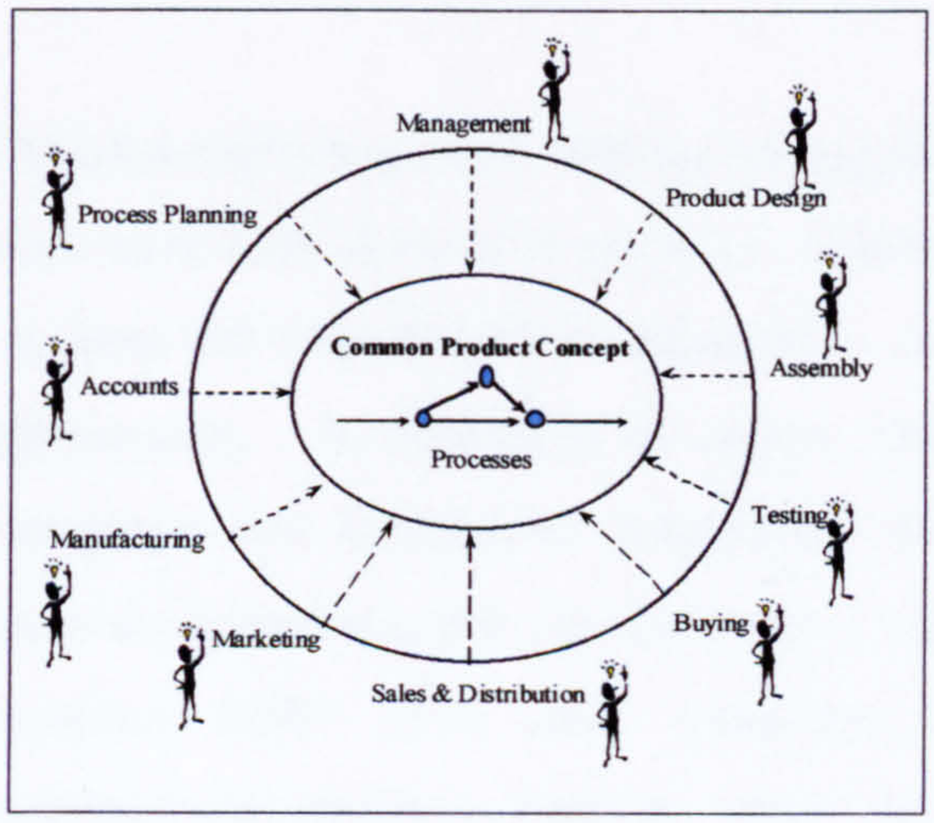


Figure 2. Concurrent Product Development

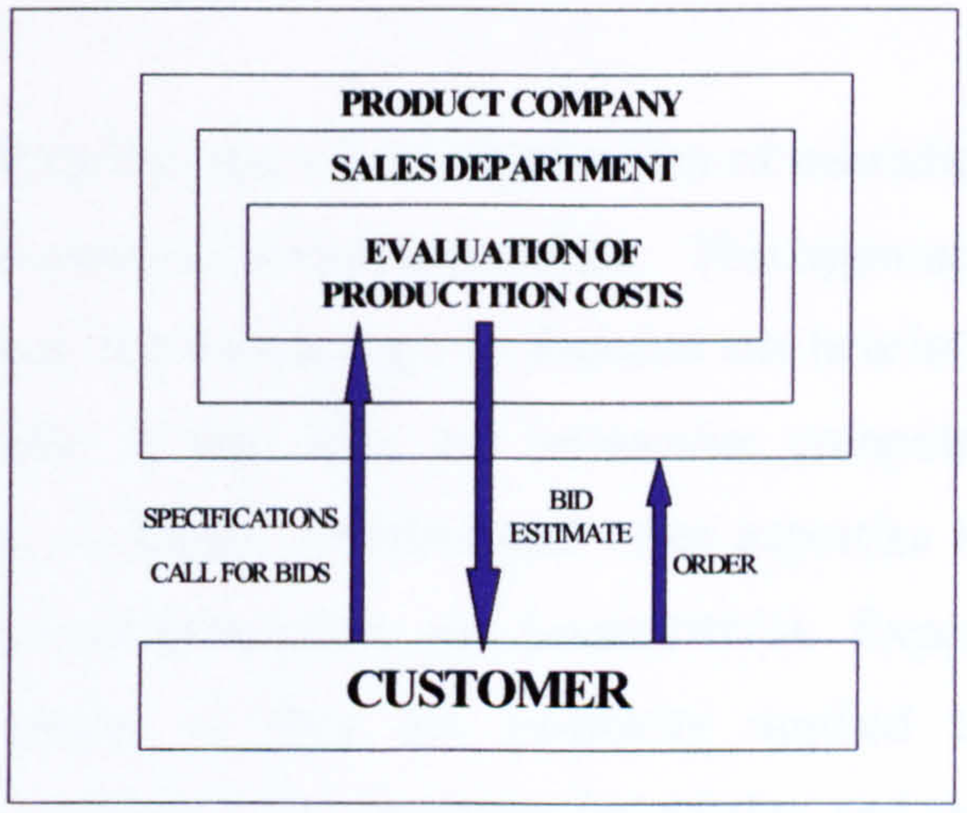


Figure 3. Cost Estimation: A key sub-process!

These risks can be minimised by the identification and reengineering of the key micro-processes within the macro-process of design. This research identifies cost estimation as one of the key micro-processes in the concurrent design framework. It does this because the quality of cost estimation directly influences the customer, firstly in the form of bid presentation and secondly in the final product cost (Figure 3). Processes that affect the customer directly automatically qualify as key *processes (Carr & Johansson(1995)*).

The current state of cost estimation is a series of dispersed, isolated departmental functions that result in difficult cost control. The reengineered state is to be a central, cohesive process that dissects the vertical functional boundaries shown in Figure 1, and can be accessed throughout the design phase.

A successfully reengineered cost estimation process would allow management to make more accurate bid estimates and encourage designers to design to cost, leading to tighter cost control while reducing the amount of design rework and product time to market.

The requirement is for costs to be modelled throughout the design phase from conceptual to detail design. Therefore, it is necessary that they be modelled without detailed process planning. Most of the research work in this area has estimated costs at the design phase using mathematical modelling and empirical formulas as the primary tools for cost estimation. (*Poli et al.(1991), Boothroyd and Reynolds(1989), Apgar and Daschbach(1987), Knight and Poli(1985), Dewhurst and Boothroyd(1987)*)

An alternative approach that has emerged in recent years is the application of heuristics that have been derived from years of design and production experience. This approach is ideal for expert system technology, since this technology is founded on heuristic processing. A definition of expert systems is that they are interactive computer programs that incorporate judgement, rules of thumb, intuition and other expertise to provide knowledgeable advice about a variety of tasks (*Dym and Levitt(1991)*). Expert systems differ from other computer systems as they are primarily applied to unstructured problem domains, where the emphasis is on symbolic knowledge and the quality of the produced solution is often judged by comparison with the human expert's solution.

Research work to date in the area of knowledge-based cost estimation has focused on the design for manufacture end of the design phase, strongly relying on a detailed design description (*Venkatachalam(1990), London et al.(1987)*). As discussed earlier, by this point it is too late to truly affect the product cost, as earlier design decisions will have decided the majority of the cost. The greatest cost benefits can be achieved by

providing designers with cost data throughout design so that they can select the right design from the conceptual design phase and further develop it for cost effectiveness. Therefore, there is a need for research in the area of knowledge-based cost estimation at the design phase, particularly before detailed design is entered into. The aim of this research work is to address this need and explore the issues associated with knowledge-based cost estimation throughout the design phase, from conceptual to detail design.

1.2 Contribution to new knowledge

The contribution to new knowledge from this research is a novel method for modelling product costs throughout the design phase, using knowledge-based methods as the central enabling technology. The designed methodology addresses the issues involved in knowledge-based cost estimation from conceptual to detail design in a framework that allows cost estimates to evolve as the design evolves. The method described is aimed at innovative design activity in which original designs bearing some similarity to past designs are produced. An architectural strategy that accommodates this method is also described.

1.3 Chapter Contents

Chapter 2 discusses cost estimation. The role of cost estimation at the different stages of design is examined. Descriptions of the various cost contributors are given and the relationships between the various contributors analysed. Parametric cost estimating methods are reviewed along with knowledge-based methods for comparison.

Chapter 3 describes the main enabling technologies utilised in this research work. These include knowledge-based systems, object-oriented methods, case-based reasoning and feature modelling.

Chapter 4 outlines the cost modelling methodology. The design of the cost modelling methodology is discussed and the reasoning behind the placing of the enabling technologies given. The cost modelling strategy is described with the aid of real world

examples selected from an industrial case study in machine design. The methodology is evaluated through a prototyping approach.

Chapter 5 details an architectural strategy that is particularly suited to the implementation of the designed cost modelling methodology.

Chapter 6 describes the Cost Expert Prototype, its development environment, its high level design, how knowledge is represented within it and how it is reasoned with. It also outlines the designed graphical user interface.

Chapter 7 outlines a plan for the validation, verification and test (VVT) of the designed methodology and reports the results of the VVT activity.

Chapter 8 concludes the thesis with a discussion of the results and an appraisal of the designed cost modelling methodology. Suggested directions for future research are also given here.

Appendix A outlines the details of the industrial case study in machine design considered in this research work.

1.4 Chapter Summary

This chapter has established the need for research and placed this research work in context. The aims of the research work are given and the contribution to new knowledge stated. A chapter by chapter walkthrough of the thesis is also included.

2 Cost Estimation

2.1 Role of Cost Estimation

Many factors have to be considered when calculating a cost estimate. Factors such as the constantly changing nature of technology, availability of materials and labour and the value of the monetary unit have to be accounted for. Cost estimates are compiled from utilising a combination of past similar product costs, established in-house cost knowledge and published cost information. Published cost information includes cost indexes which allow estimates to be adjusted to the present industrial environment, for example, *Marshall and Swift Equipment Cost Indexes* are compiled for over forty different industries and take into account the above mentioned factors.

Cost estimation falls into two categories, that of capital cost estimation and operating cost estimation. The former refers to estimating the entire cost of a project while the latter refers to the actual cost of production. This research study is limited to operating cost estimation. Operating cost estimation is crucial from a design-to-cost perspective as it assists in the following:

- Appraisal of alternative design solutions;
- Identifying potential design areas for redesign;
- Evaluation of the commercial viability of redesign efforts; and,
- Sensitivity analysis of individual components for design optimisation.

For these design considerations, a cost per unit analysis is important, and is the focus of this research study.

Figure 4 illustrates how the role of cost estimation changes as the design phase progresses.

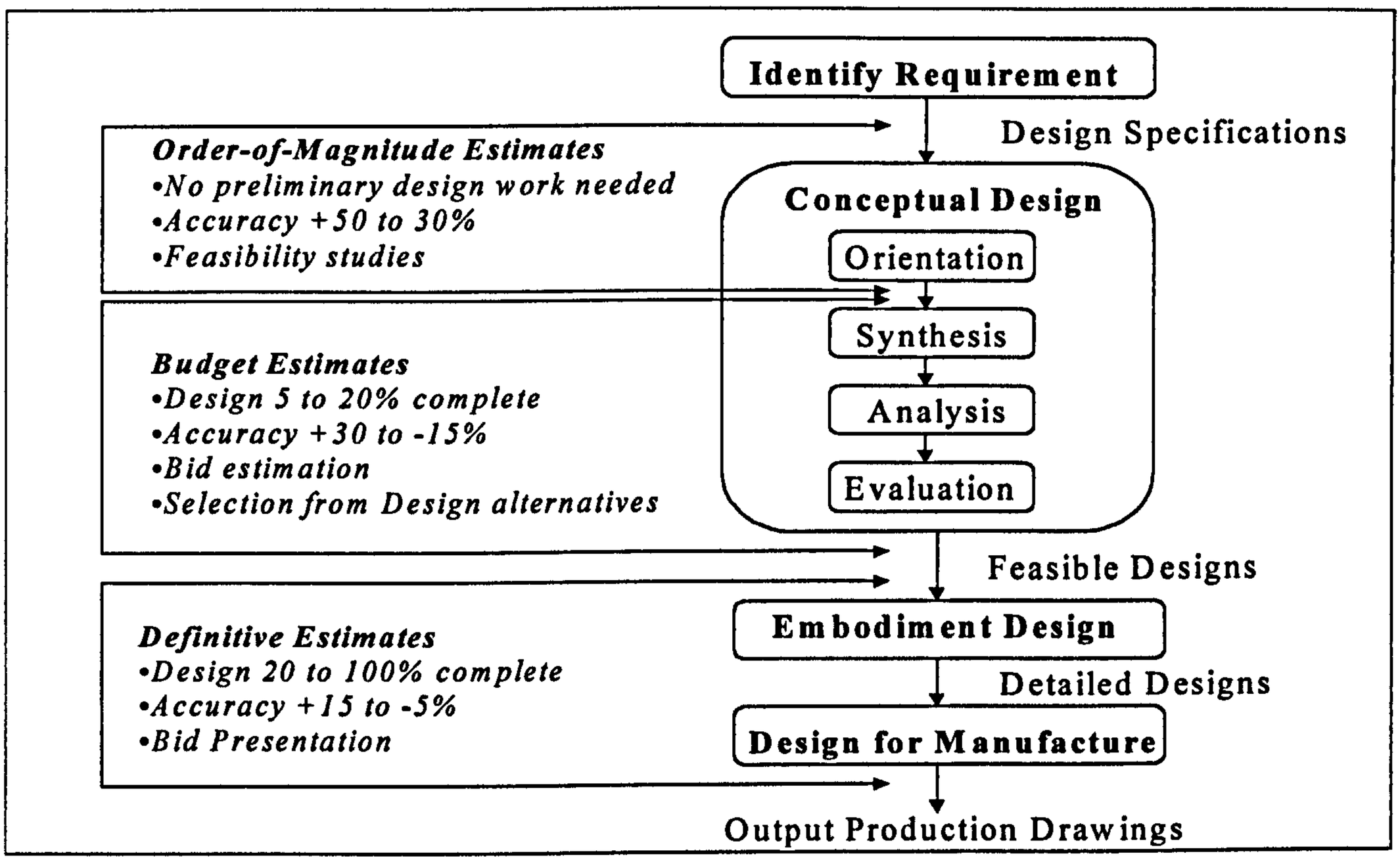


Figure 4. The Changing Role of Cost Estimation

The earliest cost estimates can be made without any preliminary design work based on the design specification and problem orientation. These types of estimates are quick and easy to produce being based simply on a cost-capacity ratio, however they are the least accurate and only useful for initial feasibility studies. The cost-capacity relationship is defined in equation (1) in which the cost of a new design is based on the cost of a similar past design and its associated capacity. A cost-capacity factor of 0.6 is recommended for use in the absence of a domain specific calculated factor (*Humphreys(1995)*).

$$Cost_{new} = Cost_{old}\left(\frac{Capacity_{new}}{Capacity_{old}}\right)^{0.6} \qquad \ldots (1)$$

The accuracy of cost estimates increases as the design content increases. At the synthesis phase of conceptual design, the accuracy of cost estimates increases to +30 to –15% (*Humphreys(1996)*). These estimates are sufficiently accurate for the purposes of selecting from alternative design solutions and bid estimation. The timely provision of these estimates would aid the designer in selecting the right design at the concept stage

and reducing the amount of design rework. Once the selected design is between 20 to 100% complete, definitive estimates can be made to a certainty of +15 to –5% (*Humphreys(1996)*). A definitive estimate can be used as a quantitative measure of the manufacturability of a design, encouraging the designer to design to cost.

2.2 Cost Contributors

Cost contributors are defined here as all the system design and planning characteristics that may contribute to the system costs. These are distinct from cost drivers on which parametric cost estimation is based. Cost drivers are defined as the system design or planning characteristics that have a prevalent effect on the system's costs.

The cost contributors for each manufacturing class are particular to that class. For example, elements that come into play in injection moulding costs will be different for those in die casting costs. Similarly, elements that come into play for manual assembly will differ from those in automatic assembly. Rather than present an exhaustive list of cost contributors that span the range of manufacturing classes, here are presented the manufacturing classes that are considered in this research study which are namely machining and manual assembly. Please refer to *Boothroyd et al. (1994)* for details on other manufacturing classes.

Figure 5 illustrates the cost contributors for the considered manufacturing classes and the dependencies between them.

It can be seen from Figure 5 that cost estimation is a complex process involving input from a diverse number of sources that are inter-related in a complex way. An example of such a complex relationship is the one that exists between material selection and process selection. Each cost contributor also contains internal dependencies. These are illustrated for the sub-contract contributor. What follows is a discussion of the main cost contributors.

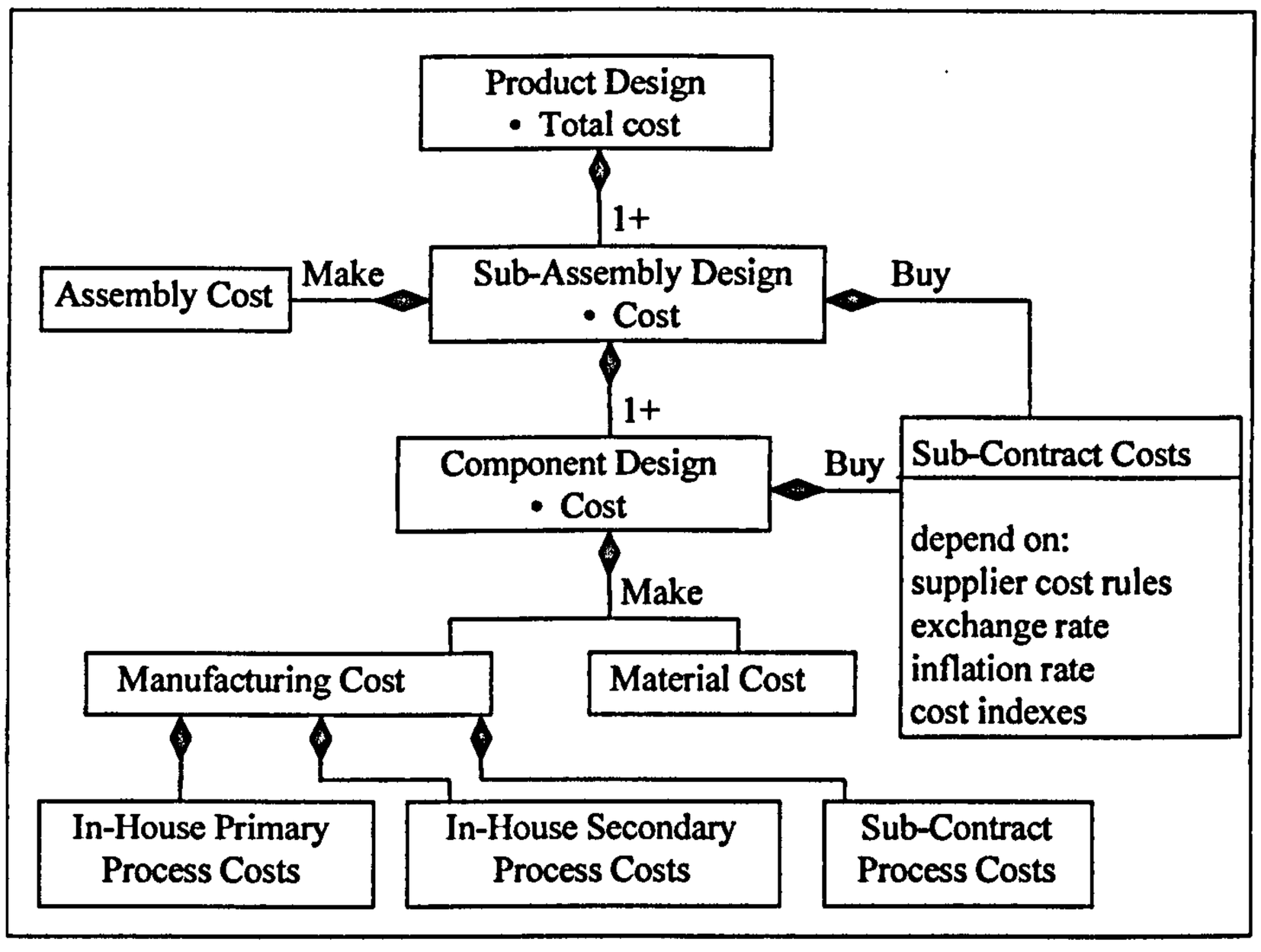


Figure 5. Cost Contributors

2.2.1 Cost Contributors for Machined Components

Component machining cost can be summarised as follows:

$$\begin{aligned} &Total\ Component\ Manufacturing\ Cost = \\ &\sum_{mach=1}^{m}(Machine\ hourly\ rate \times (\frac{Set-up\ time}{Batch\ quantity} + \sum_{op=1}^{n}(Operation\ unit\ time))) + \\ &Sub-contract\ operation\ costs + Material\ Costs + \\ &In-house\ process\ costs. \end{aligned}$$

The cost contributors for machined components are:

(a) *Material Cost.* The cost of the raw material is a very important contributor to component cost as it often accounts for more than 50% of the total cost (Boothroyd, Dewhurst and Knight(1994)). Accurate raw material costs are critical in estimating component costs. Material cost data can be obtained either directly from suppliers or from published data.

(b) *Material Class.* The specified type of raw material is a deterministic factor of Machinability and its influence on component cost has to be accounted for.

(c) *Set-up Cost.* This is the non-productive cost incurred in getting ready for machining and is determined by the set-up time which is allowed once per batch of components and should account for the following:

- Tool setting. This element relates directly to the number of tools used in the work centre operation. It includes obtaining or receiving the tools from the tool setting area or stores, checking the tools or performing any tool adjustments.

- Job preparation. This element includes receiving instructions including drawings and routings; studying the operational instructions; inspecting the "first-off" at the machine or at the inspection area; cleaning or stripping down the machine at the end of the machine cycle and loading or removing tools etc. from the machine.

- Machine setting. This element includes the fitting and setting up of special fixtures or other work holding devices such as vices, chucks, sine tables etc. Also includes input of tool data, offsets etc.

(d) *Unit Cost.* Unit (labour) time is allowed for every component in the batch. Unit time should include the following:

- Actual machining time (metal cutting),
- Machine positioning time,
- Component loading time,
- Tool change time,
- Any work alignments,
- Probing time for component orientation etc.,
- Local machining centre cleaning for work loading,
- Any interim releasing of work holding to relieve stresses induced during machining,
- In-process gauging where necessary, and,
- Any standard relaxation and contingency allowances.

(e) *Sub-Contract Operation Costs.* These are the costs incurred when a component needs to be sent to a contractor for specialist machining that may either not be within the manufacturing capability of the primary manufacturer or be more economical to sub-contract.

2.2.2 Cost Contributors for Manual Assembly

The cost contributors associated with each component in manual assembly are that of handling and insertion. Component assembly cost can be summarised as:

Assembly Cost = Labour Hourly Rate x (Handling time + Insertion time)

Factors that contribute to component handling time include the following:

- Thickness.
- Size.
- Weight.
- Symmetry. Alignment of the axis of the part that corresponds to the axis of insertion, and rotation of the part about this axis.
- Handling difficulty. Whereas the above factors are calculable, handling difficulty is a qualitative factor.

Factors that contribute to component insertion time include the follows:

- Location of the part within the assembly. If the location is obstructed or there is restricted vision then an additional time factor has to be accounted for.
- Resistance to insertion. For example, if chamfers on both mating parts have been provided then the time factor to account for would be less than if no chamfers were provided for ease of insertion.
- Ease of alignment and positioning during assembly. For example, symmetrical parts would be easier to align than partly symmetrical or asymmetric parts.

- Method of securing part within the assembly. For example, snap/press fits would be less time intensive than screw tightening.
- Ease of tool operation. If the tool has easy access and good visibility then the time penalty will not be as great as if the tool is restricted greatly by the location.

2.2.3 Cost Contributors for Sub-contracted work

A primary manufacturer may decide to buy in particular components for the same reasons as for subcontracting specialised machining operations. The proportion of bought-in components has increased substantially in recent years as manufacturers have become more focused on their own main manufacturing activities choosing to outsource any work outside their manufacturing envelope. Therefore, estimating sub-contract costs accurately has become increasingly important. Sub-contract costs can be estimated best by applying supplier cost rules. In the absence of such rules, sub-contract costs can be estimated by adjusting past product costs to the current industrial environment. The easiest way to do this is to use cost indexes. Published cost data such as cost indexes should always be treated with care as it is not often explicit how the data is derived and exactly how relevant the data is to each individual application.

2.3 Parametric Cost Estimation

Parametric cost estimation is a "top-down" approach to cost estimation in which costs are predicted for the end product, and when required, component cost breakdowns generated. Parametric cost estimation has been widely used in industry for many years as an alternative to the traditional "bottom-up" approach to cost estimation, particularly for the purposes of bid estimation. Its main advantage is that cost estimates can be produced quickly and with reasonable accuracy based on the product specification.

In parametric cost estimation, a parametric model is applied to the problem domain in order to predict the cost of the final product. A parametric model consists of a set of derived mathematical cost estimation relationships. This equation set may contain both cost-to-cost and cost-to-non-cost parameter relationships. The non-cost parameters, known as cost drivers are technical parameters particular to a family of products that

have to be identified. For example, in aircraft parametric cost models, speed, range, and altitude are often identified as the cost drivers. Parametric cost estimation can be applied throughout the product's life cycle; however, a different parametric model has to be developed for each of the different phases of the product's life.

Development of parametric cost models involves the following steps:

1. Data Collection
2. Normalisation and Calibration.
3. Identification of the cost drivers which should be confirmed with field experts.
4. Plotting cost-to-cost and cost-to non-cost data sets and testing relationships.
5. Regression and Fitting curves of best fit.
6. Data analysis and correlation.
7. Deducing the set of cost estimation relationships.
8. Validating the parametric model.

Examples of parametric models reported in the literature for predicting production costs include:

- *Dewhurst and Boothroyd (1987)* who have developed a cost model for injection moulding,
- *Knight and Poli (1985)* who have developed a cost model for forging designs, and,
- *Hoult and Lawrence (1996)* who have developed a cost model for predicting component manufacturing costs at detail design. In the cost model, component complexity is identified as a cost driver and the logarithm of dimension divided by tolerance is the metric used for measuring component complexity. Cost estimating relationships between manufacturing time and dimension information are deduced for a discrete number of machining processes and are shown to be of comparable accuracy to the human expert.

The disadvantages of parametric cost estimation are as follows:

- Difficulty of data collection.
- Reliability of parametric model outside its range.
- Accuracy of specified values of cost drivers.

- Narrow domain-dependency of parametric models.
- Development of the parametric model itself.

2.4 Knowledge-Based Cost Estimation (KBCE)

Knowledge-based cost estimation is an alternative to parametric cost estimation and is the focus of this research study. Rather than involving a statistical analysis of historical data, KBCE involves the capture of cost estimating expertise. The advantage that KBCE has over parametric cost estimation is that it can be easily applied in domains where extensive historical data is not readily available. In addition, the capture of domain knowledge is less time intensive than the data collection and analysis of historical data. KBCE however is not exclusive, where appropriate developed and tested cost models can also be incorporated as domain knowledge. This is especially relevant when attempting to model costs throughout the design phase. Unlike parametric cost estimation, in which separate parametric models have to be developed for the different phases of design. In a knowledge-based framework the actual cost estimating approach can change between one phase of design and the next, depending on availability of costing sources. Both parametric and knowledge-based cost estimation rely on past experience, the difference is that the former models this past experience algorithmically while the latter models it symbolically.

Application of expert systems to cost estimation is a relatively new approach hence the majority of development and research work in this area has been in academia. These research efforts have mainly focused on the design-for-manufacture (DFM) end of the design cycle. The results of such efforts are prototype systems, whose central aim is to establish the principles and supporting techniques for the application of expert systems in DFM. Examples given in the literature for such work include:

- *Venkatachalam et al. (1991)* who present a framework for implementing the DFM approach using KBS methodology. The methodology presumes that a detail design description is available and the user is prompted to input this description manually.

The method processes this input, by applying domain cost estimation rules, and outputs a cost per unit.

- *Taylor (1997)* who presents a knowledge-based approach to cost estimation in which designs are described as feature sets. These feature sets are used to drive the cost estimation process.
- *London et al. (1987)* who present a expert system shell that can be readily customised to incorporate cost estimation rules for each different manufacturing class. Again, a complete design description is assumed.

More recently, focus has shifted to modelling costs at the earlier stages of design, however the research reported has always resorted to the development of parametric models.

2.5 Chapter Summary

This chapter discusses the process of cost estimation and the role cost estimation plays throughout the design phase. The class of cost estimation considered in this research is also stated. This research is concerned with operating cost per unit analysis. The cost contributors to be modelled are also outlined here. The two alternative approaches to cost estimation are introduced, that of parametric cost estimation and KBCE. Both parametric cost estimation and KBCE have pros and cons attached to them. The choice of approach is very much situation dependent. Parametric cost estimation relies heavily on historical data sources, therefore effective data collection is very important. It is also extremely time consuming. In the absence of extensive data, KBCE would be the better choice. On the other hand, KBCE relies on knowledge elicitation from domain experts. In the absence of experts that can communicate their expertise well, parametric cost estimation may be the better option. Where good experts are available KBCE can prove to be a very effective approach.

3 Enabling Technology

3.1 Knowledge-Based Systems (KBS)

'Artificial Intelligence is that branch of computer science dealing with symbolic non-algorithmic methods of problem solving (*Buchanan (1984)*). Humans have the mental ability to manipulate both concepts and numbers. They are able to solve problems through investigation as well as by following procedures. Artificial Intelligence (AI) is a science that attempts to simulate intelligent human behaviour on computers.

One of the main branches of AI is that of Knowledge-Based or Expert Systems. 'A knowledge-based expert system is a computer program that performs a task normally done by an expert and which, in so doing, uses captured, heuristic knowledge (*Dym and Levitt (1991)*). Heuristic knowledge is knowledge that guides to a solution without the exhaustive application of an algorithm. Heuristic knowledge can be in many forms, examples include experiential rules of thumb, high level knowledge about how to use other kinds of knowledge and rules describing fundamental principles. The main issues that need to be addressed in KBS are that of knowledge acquisition, knowledge representation, reasoning and explanation.

3.1.1 Knowledge Acquisition (KA)

The Knowledge acquisition process involves the obtaining of the problem domain heuristics from the domain experts. KA is widely regarded as the bottleneck to developing applications. This is because very often experts find it difficult to impart their knowledge and this can be for a number of reasons such as time constraints, lack of commitment and difficulty experienced by experts in trying to express their knowledge. A range of techniques exists to aid KA such as interviewing techniques, task analysis and Kelly grids. A detailed description of these methods can be found in *Hart (1989)*.

3.1.2 Knowledge Representation (KR)

Problem domain knowledge has to be represented in a manner appropriate for computational processing. The main representation paradigms are as follows:

- *Procedural Representation* attempts to model the problem domain as a discrete process. Its application is limited by its lack of expressive power.

- *Rule-Based Representation* is based on the logic-programming paradigm. Here the problem domain is decomposed into rule sets. The rules themselves are usually expressed as *If-Then* structures. This is a very useful construct as a substantial amount of expert reasoning can be easily expressed in the form of situation-action like rules. However, rules lack expressive power when employed in complex domains such as Design and Engineering where abstract concepts and relationships between these concepts may need to be modelled.

- *Semantic Nets* are a rich modelling paradigm as they can be used to model almost any relationship that may exist between problem domain entities. A semantic net is a formal graphic language for modelling. It is drawn as a labelled, directed network in which labels on nodes correspond to physical or abstract objects and their properties, while labels on arrows that connect the nodes represent entity-entity relationships. The problem with semantic nets is that they try to model all entity relationships graphically and for even small problem domains, these networks can become very cluttered and difficult to manage.

- *Frame-Based Representation* is a subset of semantic nets. Frames have all the rich expressive power of semantic nets; however, they use encapsulation or information hiding to manage the domain model. All entity relationships except for inheritance and aggregation are captured within a frame. So all that needs to be modelled graphically are the abstraction structures. This allows large problem domains to be modelled and manipulated easily.

- *Hybrid KR* integrates rules, frames and procedures in order to exploit the merits and overcome the limitations of each of the modelling paradigms. This enables a large range of complex problem domains to be modelled effectively.

3.1.3 Reasoning

In KBS, the inference engine is responsible for reasoning. Its function is to access and manipulate the knowledge base in order to deduce solutions for the specific problem data that it is given. The inference engine essentially acts as the "Brain" of the expert system. It uses reasoning methods, such as forward chaining, backward chaining and bi-directional chaining, to organise the correct sequence of heuristic activation. These reasoning methods are all examples of directed search as opposed to blind search. These search methods are discussed in what follows.

Search methods can be used for problem solving, exploring alternatives and testing/proving theories and techniques. Given an initial state and a goal state, search methods apply operators that embody the relationships between the various states to obtain successor states. These are checked against a goal state to see if a solution has been reached. Search methods are divided into two categories that of blind search and directed search. Blind search methods are ones in which every single node in a search space is expanded until a solution is reached. These methods are computationally intensive and referred to as weak methods. Breadth-First and Depth-First search are two examples of blind search.

Directed search methods are those in which the search is knowledge-guided and these methods are referred to as strong methods since they are much more efficient than blind methods. In KBS, deduction and search complement each other in representing and reasoning with knowledge. Directed search methods employed for deduction in KBS are as follows:

- Forward chaining or data-driven reasoning is one in which search is performed forward from the initial state by applying the modus ponens rule of inference, which is, 'Given A and A → B; deduce B'.

- Backward chaining or goal-driven reasoning is one in which search is performed backward from the goal state applying the inverse of the modus ponens rule.

- Bi-directional chaining is when one can reason both forward from the data and backward from the goal depending on the best action to take at that time.

The reader is referred to *Gonzalez and Dankel (1993)* for a detailed description of the search methods outlined above.

3.1.4 Explanation

Provision of explanation for any solution produced by a KBS is necessary if a KBS is to gain user acceptance. It is also useful to the knowledge engineer for validation purposes. Methods that facilitate explanation include rule traces and transparency of reasoning. Figure 6 illustrates the basic components of a KBS.

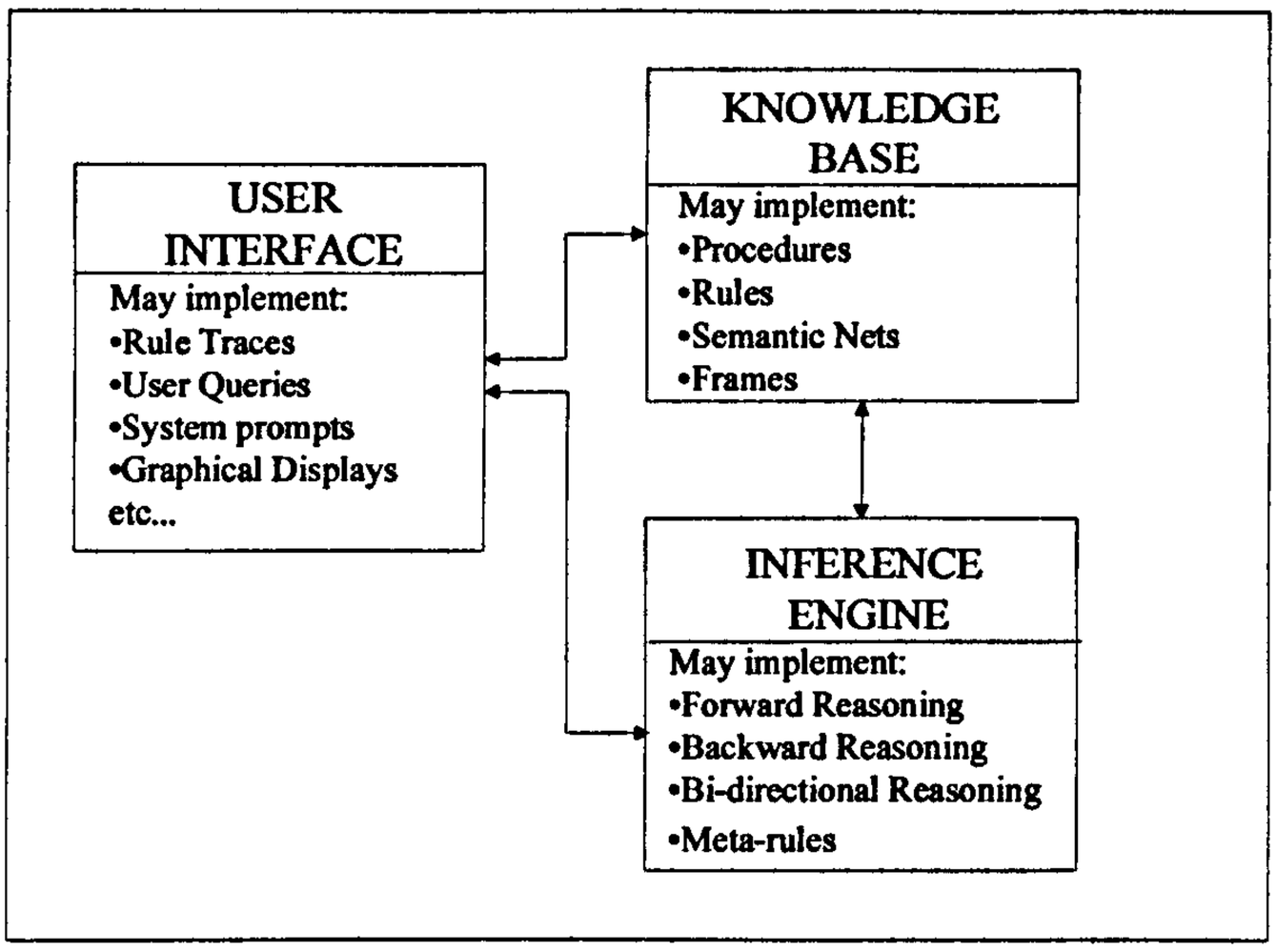


Figure 6. Basic KBS Architecture

3.2 Object-Oriented Methods (OOM)

Object-oriented methods are closely related to both semantic nets and frame-based representations. They build on the rich modelling paradigm of semantic nets by incorporating the modelling concepts of abstraction and encapsulation into an object-modelling paradigm. Objects are very similar to frames in that they model abstract entities by collecting data related to that entity and encapsulating it within an object. However, object models do not encapsulate the relationships that may exist between objects, they only encapsulate the object's attributes. Unlike frames objects also encapsulate the behaviour associated with them.

The concept of abstraction is what allows object models to be as expressive as semantic nets while at the same time being as easily manageable as frame-based representations. Abstraction allows the same problem domain to be represented and viewed at different levels of detail. Therefore, object-oriented methods are ideal for modelling complex and/or large problem domains.

From a KBS development perspective, an object-oriented analysis of the problem domain could easily be decomposed into the associated frames, rules and procedures at the system design phase.

3.2.1 An Object-Oriented Method for Analysis

What follows is a description of the object-oriented method used in this research for analysing the cost estimating problem domain. The method employs the widely used Object Modelling Technique, OMT (*Rumbaugh et al. (1991)*) notation extended with SOMA constructs. SOMA is an OOM filter proposed by *Graham (1994)*. In Soma, the best ideas from the range of OOA methods available are combined to produce a semantically rich framework for OOA. The main extensions to OMT are:

- Subjects. SOMA borrows the construct of subjects from the OOA and OOD methods outlined by *Coad & Yourdon (1991)*, who use subjects to manage complex and large problem domains.

- Rules. SOMA adds rules to the class and object definitions. This is particularly useful for knowledge-based systems development and for applications in which control is distributed.

The approach taken by SOMA proceeds in seven phases, each phase analyses the problem domain from a different perspective. The analysis results obtained are documented in a seven-layer object-oriented analysis model. The activities that form the layers are:

1. Identify Subjects. This involves decomposing the problem domain into parts of a manageable size. Subjects give an overview of the analysis model and are useful in dealing with problem scale. The subject layers can be expanded to reveal the collection of objects they encompass. They can be further expanded to show the structures in the problem domain.
2. Identify Objects. This involves defining the system abstractions that reflect the problem domain entities that need to be modelled.
3. Identify Structures. Structures model the real world constructs that may exist between the identified objects. The constructs looked for at this phase are those of classification, composition and usage.
4. Define Data Semantics and Associations.
5. Add Attributes. Attributes of objects represent the properties associated with the real-world objects; i.e. they represent the object state.
6. Add Operations. Operations associated with the identified objects are defined. Operations describe the state-changing behaviour of the objects.
7. Add Rules. Rules are distinguished from operations in this method, in order to allow a smooth transition from analysis to design. Rules are added to express the constraints on the state changing behaviour of the objects. Rules themselves can also be state changing.

The complete modelling notation is illustrated in the notation section.

3.3 Case-Based Reasoning (CBR)

Case-based reasoning is reasoning by analogy. It is similar to knowledge-based reasoning as both capture past experience and utilise it for problem solving. The difference is in the manner in which this past experience is captured, represented and utilised. Knowledge-based reasoning captures past experience in the form of domain heuristics and utilises the captured knowledge by employing a selected inference mechanism for problem solving. Case-based reasoning on the other hand captures past experience in the form of specific problem-solving episodes and utilises the captured knowledge by employing a suitable case adaptation strategy. The choice of reasoning paradigm is situation dependent. Case-based reasoning is a more appropriate paradigm when problem domain knowledge cannot be easily represented as a discrete set of generalised heuristics. In addition, case-based reasoning is the method of choice when problem solving is most easily accomplished by recalling a similar situation in the past.

Applications that have employed case-based reasoning range from meal design to medical diagnosis. Examples of such applications include CASEY (*Koton (1988)*), a heart failure diagnostic tool; JULIA (*Hinrichs and Kolodner (1991)*), a meal designer; CHEF (*Hammond(1986)*), a Szechwan cooking planner; ARCHIE (*Goel and Kolodner (1991)*), an architectural design aid; and many more. CBR in diagnostic applications is more mature than in design applications. This is because of the complex nature of design. The main application of CBR in design has been in the form of design aids such as CASECAD (*Maher and Balachandran (1994b)*), a structural design aid and FABEL (*Vob et al. (1994)*), a building design aid; rather than design synthesis. There are however many research efforts reported in the area of design synthesis such as CADET (*Sycara and Navinchandra (1992)*), a mechanical design solver; CADSYN (*Maher and Zhang (1993)*), a structural design solver; and CYCLOPS (*Navinchandra (1988)*), a landscape designer.

The main issues that need to be addressed in implementing CBR are that of case collection, case representation, case recall and case adaptation. Here we are considering utilising a CBR facility in the domain of design and what follows is a brief discussion of the identified main issues from a design perspective. For a more detailed treatment of

these issues, the reader is referred to *Maher, Balachandran and Zhang (1995)*. The reader is also referred to *Kolodner (1993)* for a more general and in depth discussion of CBR as a problem-solving paradigm.

3.3.1 Case Collection

Design case collection, like the knowledge acquisition process in KBS is driven by the domain expert. The expert has to guide to what the content of the design cases needs to be in light of the function the case-based reasoning facility is to serve. Once the content of the design cases is decided than case collection itself can either be:

- Automated in which the content of the design cases is assumed to be already stored electronically in a CAD environment and the design cases are defined by some form of transformation of the currently stored designs;
- Interactive in which the designer enters the cases into the case base online as in an automated knowledge acquisition facility; or,
- Manual in which the cases are elicited through traditional knowledge acquisition means such as interviews and task analysis. This is often the only option if the case content specified is not made explicit in a company's data sources.

3.3.2 Case Representation

The selection of a representation schema for design cases is strongly influenced by the case content. Case content determines the context and the effectiveness of a CBR facility to meet its requirements specification. Design case content may be in the form of design drawings, requirements and solution sets, function-behaviour-structure definitions, or, a combination of these. Once the case content is defined, the issue of case representation that is central to any CBR facility needs to be resolved. Commonly implemented forms of design case representation include:

- Design models which are abstractions of particular groupings of design cases that may incorporate any combination of the other representation schemes;
- Attribute-value pairs which describe the design in terms of key-value pairs that have some semantic value;

- Textual descriptions which describe a design in the form of a report, of the associated design activity, that may contain notes on the problem being addressed and a trace of the design decisions made;
- Object-Oriented representations that describe a design by associating it with a design type. The concepts of abstraction and aggregation may also be applied in such design representations;
- Graph-based representations which describe a design by modelling dependencies between the design's features;
- Multimedia representations which describe designs by incorporating different representations such as text and graphics into the design case;
- Cases/Subcases Hierarchies which describe a design in terms of its whole and parts; or,
- A combination of the above.

3.3.3 Case Recall

Once the representation schema for cases has been decided, the cases have to be organised in case memory so that they can be recalled easily when required. The recall protocol has to specify how cases are indexed, retrieved and selected. The main schemes used for design case indexing are:

- Flat structure indexing, in which the cases are indexed by a single unique identifier, most commonly being the name of the design;
- Feature-based indexing, in which cases are indexed by multiple identifiers, for example, in a Function-Behaviour-Structure case representation, a number of attributes may be specified as keys to the case; and,
- Indexing trees, in which cases are indexed by a hierarchy of identifiers. These are used particularly when some sort of a case hierarchy is imposed on the case base.

The main schemes used for design case retrieval are:

- List-checking, which is simple surface feature matching in which case memory is searched and the cases matching a given list of features is retrieved;
- Concept refinement, which utilises index trees. This is again simple feature matching, however here case memory is searched incrementally starting at the top of

the indexing hierarchy with a list of top-level matching features to match, search progresses down the hierarchy only if a match is found at the higher level; and,

- Associative retrieval, which involves the retrieval of the most appropriate cases rather than the closest matching cases. This is a combination of surface feature match and deep feature matching. That is when a set of matching cases is retrieved, it is further reduced by applying relevant domain knowledge.

The main methods for design case selection are:

- Feature matching, in which the case with the highest number of matching features is selected as the best case;
- Weighted feature matching, in which weights can be assigned to case features indicating the relative importance of a feature match; and,
- Context-dependent matching, in which domain knowledge is applied in selecting the best case.

3.3.4 Case Adaptation

Automating case adaptation is a complex task for which definitive solutions are still to be defined. Schemes for adaptation fall in general under the following three classes:

- Adaptation by substitution. This is the simplest form of case adaptation in which any attribute values in the selected case that do not match the new problem specification are simply replaced by the attribute values defined in the specification. Also if the attribute-value pair itself does not exist, it is simply inserted into the case;
- Adaptation by derivational analogy in which the problem-solving process used in the previous design case is recalled and replayed for synthesising a solution given a new specification; and,
- Adaptation by transformational analogy in which the design solution in a previous case is recalled and changed, by application of specific domain knowledge, to meet the new specification.

3.4 Feature Recognition

Feature recognition methods have been developed in recent years to meet the requirement for richer product models posed by integrated product and process development. The aim of these methods is to transform geometric models from CAD solid modellers into feature models. Feature models are high-level, context driven product models. That is, a feature model describes a product in terms of features that are defined particularly for the end user application. For example, for a manufacturing application a product may be defined in terms of machining features while for a design application it may be defined in terms of design features. The two feature models although derived from the same geometric model and describing the same product are very different. The first step therefore in feature recognition is to specify the feature definition set for a particular application. International standards for application-specific product definitions exist in the form of STEP (ISO 10303), the International Standard for the Exchange of Product Model Data. Industry-specific application protocols (AP) are defined within STEP. Examples include AP203 – Configuration controlled 3D assemblies and AP224 – Mechanical product definition for process planning using machining features.

Feature recognition methods are classified according to the type of geometric model on which they operate. The different geometric models utilised in CAD solid modelling are as follows:

- *Constructive Solid Geometry (CSG) Models.* These store the solid geometry in the form of a construction history tree that has a binary tree structure. The binary tree structure comes about because the geometry construction scheme used involves the successive application of Boolean operators such as union, difference and intersection. These operators operate on a pre-defined set of generic shape primitives. The leaves of the binary tree are therefore primitive shapes while the intermediate nodes represent the operator being applied and the resulting solid state. The advantages of CSG models are that they are guaranteed to model only valid solids and that their data structures are concise even for complex solids. The

disadvantages are that CSG models are not unique and that their expressive power is constrained by their primitive set.

- *Boundary (B-rep) Models.* These store the solid geometry as a collection of faces, edges and vertices together with their topological definition. The advantages of B-rep models are that they allow for high quality surface representations and that they can be easily created from a wide range of construction methods. The disadvantages are that B-rep models are large for all but the simplest of solids and they are difficult to validate.

- *Sweep Models.* These store the solid geometry as a set of closed profiles with their respective sweep information. Sweep operations are in the main that of extrusion, rotation and translation. The advantages of sweep models are that they are easily stored. They allow for easy construction of complex solids from which boundary models can be easily derived.

Shah and Mantyla (1995) present a comprehensive literature survey of available feature recognition methods. Summarising, feature recognition can be facilitated by a design-by-features approach or via automatic feature recognition. The design-by-features approach has been widely adopted by CAD vendors as it provides an easily controlled design environment in which the designer is limited to designing with pre-defined feature sets. However, from a designer's perspective this environment can prove to be very inhibiting. The alternative approach is that of automatic feature recognition in which the designer is free to design as he wishes, the responsibility of creating a feature model being left to the feature recognition algorithm being employed.

Feature recognition algorithms operate on the geometric model by searching for pre-defined features and output a feature model of the design. These algorithms are classified as either pure or hybrid. Pure algorithms are either rule-based, graph based or volume decomposition based and act on either the B-rep model or the CSG model.

The main advantage of pure rule-based algorithms are that they can be written specifically to identify the features of interest and if rules are written well, they can provide successful and unambiguous recognition of features of interest (*Henderson et al. (1994)*). Examples of pure rule-based algorithms reported in the literature include *Choi (1984)*, *Vandenbrande (1990)* and *Kyprianou (1980)*. The disadvantages of pure rule-based algorithms are that:

- Rules are ad hoc in nature;
- Rule-based feature representations are not unique in that the same feature can be recognised by different rule forms;
- Verifying rule-based algorithms for completeness is an almost impossible task; and,
- Rule-based feature recognition involves repeated blind search of the solid model. This can be computationally very costly for complicated parts.

The main advantage of pure graph-based algorithms is that graph theory is well understood and established. Examples reported in the literature of pure graph based algorithms include *Joshi & Chang(1988) and Chuang(1991)*. The disadvantages of pure graph based algorithms are that they:

- Are unable to search for inexact features;
- are unable to recognise partial features or incomplete features; and,
- like rule-based algorithms involve repeated blind search of the solid model for matching graph patterns. This is computationally costly for all but the simplest of parts.

The main advantage of pure volume decomposition based algorithms is the ability to recognise all features including inexact, partial or incomplete features. Examples reported in the literature include *Sakurai and Chin(1994)*. The disadvantages are that these methods are still immature and in their current state of development limited to mainly simple prismatic parts. They are also severely handicapped by their computational cost.

Hybrid algorithms have emerged as an answer to the limitations posed by pure algorithms. These algorithms define some combination of the pure algorithmic

approaches and may specify a requirement for a number of geometric models to be accessible. Examples of hybrid algorithms reported in the literature include *Vandenbrande & Requicha(1994)* and *Regli and Nau(1993).*

3.5 Chapter Summary

This chapter has introduced the enabling technologies used in the design of the cost modelling methodology in chapter 4 and should be read in conjunction with that chapter. The enabling technologies of KBS, OOM, CBR and FR are discussed here. Justification of their placement in the cost-modelling framework is given in the next chapter.

---

4 The Cost Modelling Methodology

4.1 Problem Statement

The designed methodology has to identify and address the issues raised in automating the cost estimation process throughout the design phase, from conceptual to detail design, using knowledge-based methods.

The methodology is to be aimed at innovative design, which encompasses the majority of industrial design activity. Innovative design is defined as "design in which new variables or features are introduced, which still bear some resemblance to existing variables or features and the decomposition of the problem is known. However, the sub-problems and the various alternatives to their solution must be synthesised. In other situations, alternative recombination of the sub-problems may yield new designs. It is also considered that solving the same problem in different ways, or different problems in the same way (by analogy), would fall under this class" (*Evbuomwan et al.(1996)*).

The nature of the cost estimation process can be examined through observation. Appendix A outlines the industrial case study in machine design that is considered in this research. Observation reveals that the actual cost estimation process is ill structured and complex. This is inherent in the expert's description of the costing process. Given a design assembly, the expert takes a bottom-up approach by considering single component manufacture and constructing rough process plans for the designs and then attaching assembly costs. The cost estimates are constructed by examining the product design and applying heuristic knowledge gained from years of experience in estimating product manufacturing and assembly costs. At the early design phases, the expert given the abstract design description attempts to fill in the missing design details from past design experience and then follows the same costing process as described.

4.2 The Cost Estimation Strategy

From the simple description of how a domain expert constructs cost estimates given above, the key issues that need to be addressed when automating cost estimation throughout the design phase are identified as:

- ***Design Completion.*** A design completion facility is required if the method is to address cost modelling at the earlier design phases.

- ***Design interpretation.*** Effective design interpretation is a prerequisite for any successful cost modelling method. Without it the cost modeller would hinder the designer, with extensive prompting for design information, rather than aid him.

- ***Capture and application of cost heuristics.*** The quality of any cost estimate is directly dependent on the quality and completeness of the cost heuristics. Successful capture and restructuring of these heuristics for use in an automated process requires careful consideration.

Each of these issues deals with different levels of abstraction of the problem domain. Design interpretation deals with the lowest level of abstraction as it acts mainly on the CAD representation of a design. Design completion acts on a higher level of abstraction as it operates on the transformed application-specific representation of the design. Finally, cost heuristics are applied at the highest level of abstraction as they deal with the completed design description. As there is a natural separation of concern between these issues, the cost estimation strategy adopted is a modular one as illustrated in Figure 7. Such a modular approach is advantageous as it:

- Eases the handling of the problem domain complexity;
- Allows for modularising the validation, verification and test process; and,
- Introduces a plug 'n' play effect in which the implementation of any module can be replaced by a completely different implementation without affecting the rest of the infrastructure.

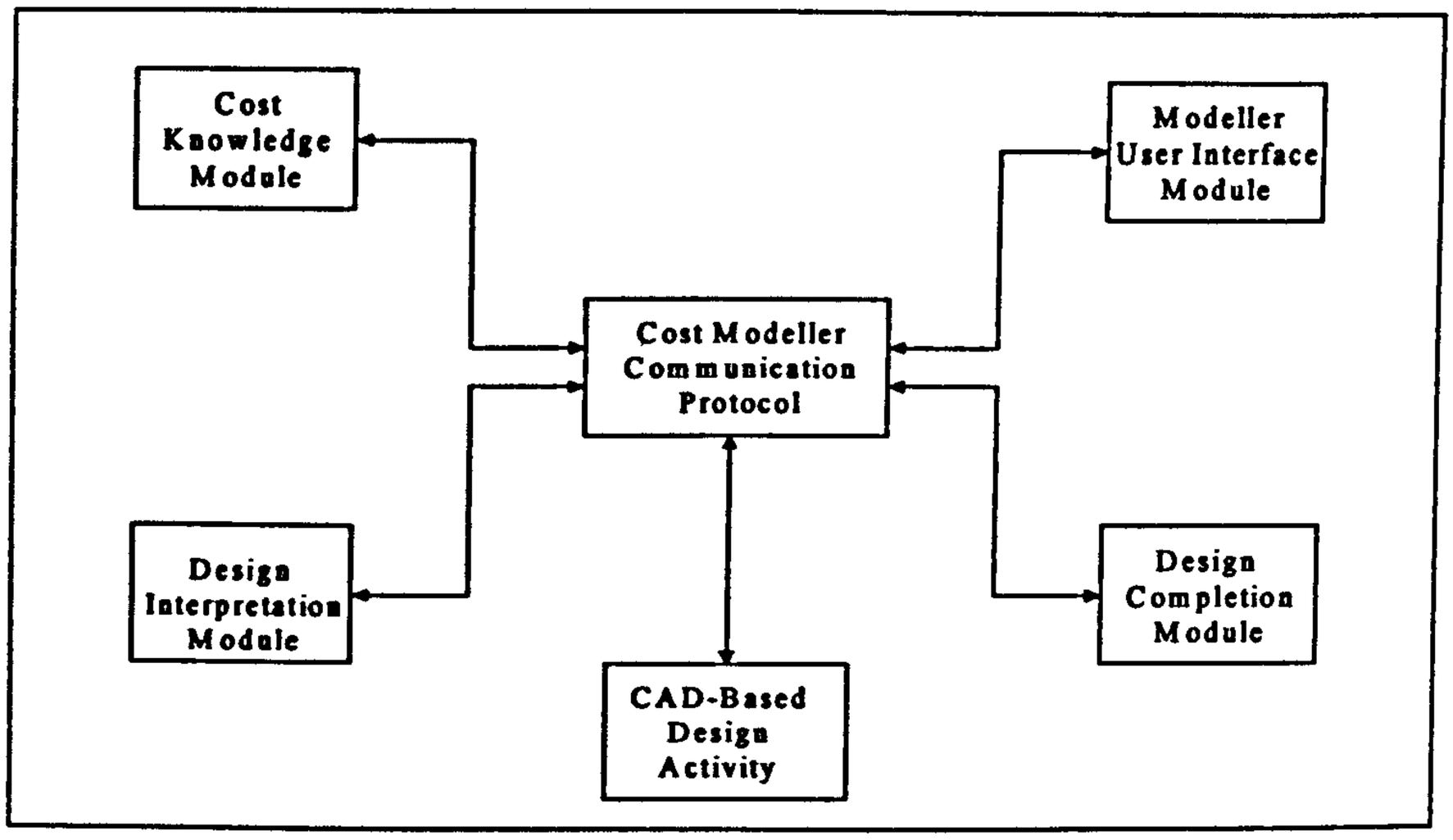


Figure 7. Modular Infrastructure Imposed by Cost Modelling Strategy

Modelling cost estimation is a complicated process, and the output from one module cannot be easily defined as the input to another module. Therefore, the modular infrastructure imposed by the cost modelling strategy must incorporate some form of a communication protocol as illustrated in Figure 7. Not only are interactions between modules unlikely to be in a simple ordered fashion but so are the interactions within modules. Therefore, all interactions in the cost modeller must be allowed to subscribe to the communication protocol. The responsibilities of the communication protocol are to:

- Allow the services offered by each module to register interest in any events that may be generated during the cost modelling process.
- Notify services of any events generated that they may have registered an interest in.
- Control the cost modelling process, i.e. the invocation of services and the order of invocation.

The communication protocol adopted by the cost modelling strategy is that of a blackboard system and is discussed in detail in Chapter 5.

What follows is a discussion of how the key issues of design completion, design interpretation and the capture and application of cost heuristics are treated in the cost modelling methodology.

4.3 Design Completion

Some form of design completion is necessary if knowledge-based cost estimation is to be employed at the earlier design stages. This approach mimics the expert's problem-solving process, who given an incomplete design description attempts to complete this description by recalling similar past designs and then reasoning by analogy in order to predict missing design and manufacturing features. Case-based reasoning is a problem-solving paradigm that provides a framework for reasoning by analogy. It is incorporated in the designed methodology in order to implement design completion. The objective of the case-based design facility is to consider the incomplete description of the new design problem, retrieve a similar past design from the case base and adapt the retrieved design to satisfy the new problem description.

The design domain under consideration governs the case-based approach taken. Here, we are concerned with the domain of machine design. Considering the nature of this design domain, the case-based approach taken here is similar to that taken by KRITIK (*Goel et al.(1997)*) and CADET (*Narasimhan et al.(1997)*) who both consider machine design. KRITIK and CADET use case-based reasoning to adapt past designs to meet new functional specifications. The case-based facility defined here does not take responsibility for case adaptation to meet functional specifications or for propagation of design changes from one part of the design to the rest of the design. That responsibility is left to the designer. The role of the case-based facility defined here is to aid in the evaluation of the produced designs from a cost perspective. *Domeshek and Kolodner(1997)* highlight the importance of design proposal critiquing and evaluation and the lack of availability of tools that work within the case-based reasoning framework. Here, the cost estimation methodology is designed such that the designer could be replaced by a case-based design facility that assumes the designer's responsibilities. The aim is to define a case-based reasoning facility that as well as aiding the cost evaluator fits in naturally with the generic case-based design facility envisaged by current research trends such as KRITIK2 and CADET. The role that case-based reasoning plays in the cost modelling strategy is illustrated in Figure 8.

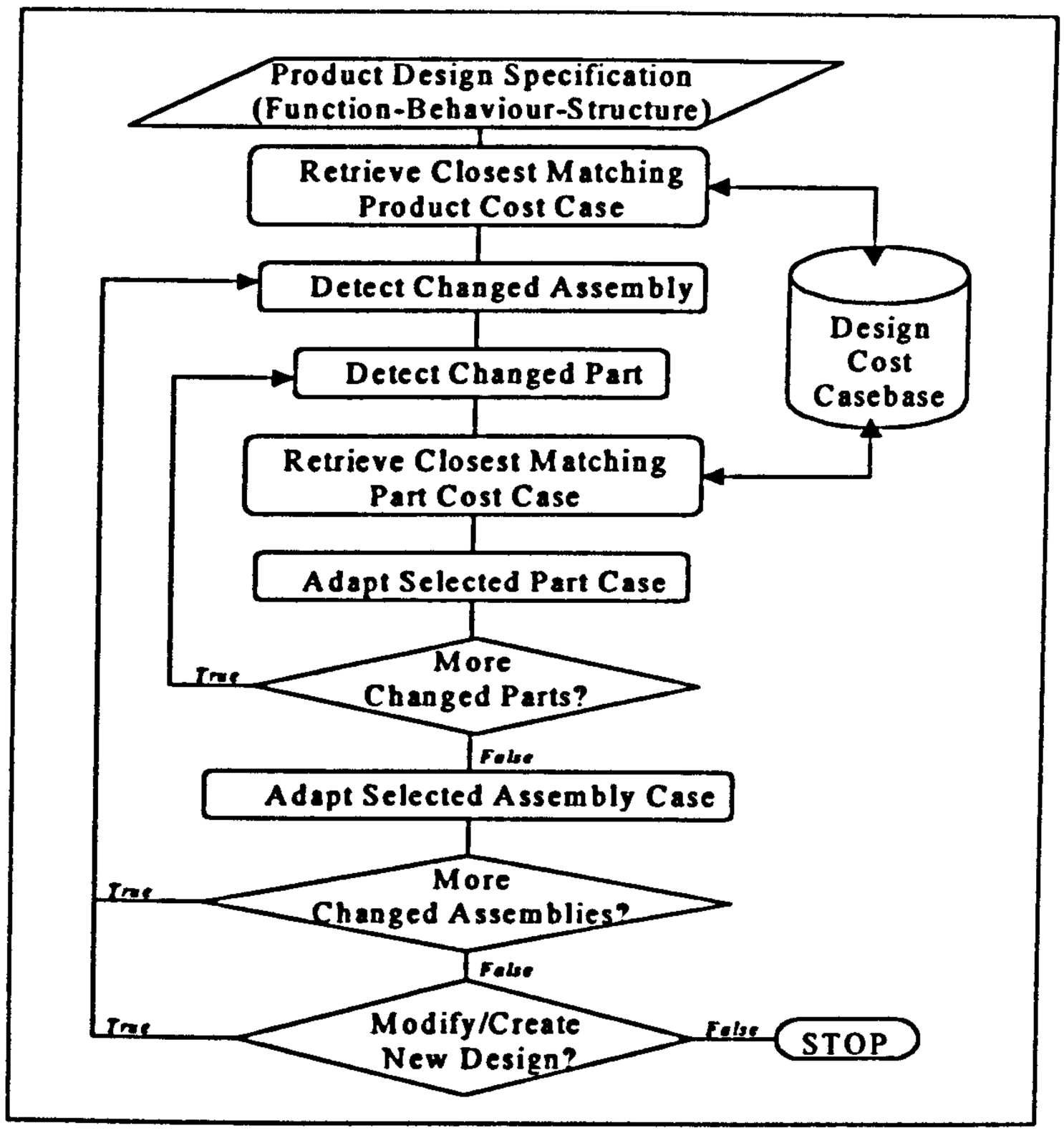


Figure 8. The Cost Modelling Strategy

What follows are details of how design cost cases are represented, recalled and adapted in the above framework.

4.3.1 Representing Design Cost Cases

Here, we are considering innovative design activity. Innovative design involves the use of past design experiences to synthesise new designs. As the decomposition of the problem is often known in innovative designs, complete past product designs can be used as blueprints for new design situations. Complete machine designs typically contain many assemblies which themselves may contain many sub-assemblies and individual parts. Presenting and managing machine designs as single cases may prove to be a difficult and unruly task. *Tsatsoulis and Alexander(1997)* present a method for integrating cases, sub-cases and generic prototypes for design. A similar approach is taken here for managing machine designs that are decomposed into multiple cases in the case-based reasoning framework.

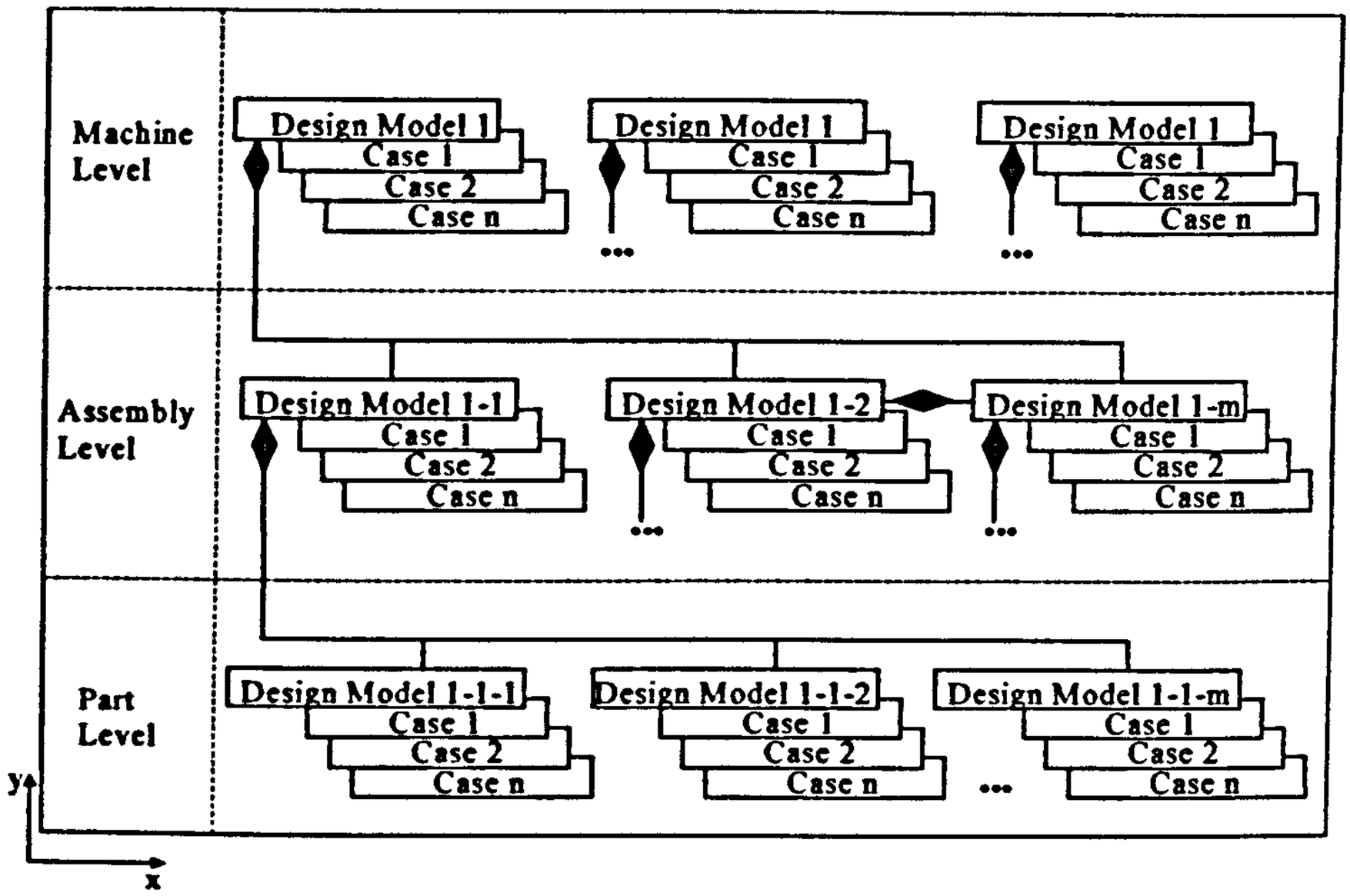


Figure 9. Casebase Structure.

An object-oriented approach is taken in defining the case-based reasoning facility. The case-base structure imposed is illustrated in Figure 9. Case memory is organised in a 3-dimensional hierarchical manner. Such a structure is imposed on case memory in order to allow multiple abstraction level browsing. Therefore, the case base can be traversed in all three dimensions. This enables sub-problems and the various alternatives to their solutions to be recalled.

In Figure 9, the x-y plane hierarchy is a whole-part one, in which each successive level represents a different abstraction level of product design; the highest level being the product design and the lowest one being the part level. Assembly and multiple sub-assembly levels reside in between the two. The x-z plane hierarchy is an inheritance one, in which design cost models are used to describe a class of design cost cases. The actual instances of design cost cases can be visualised as residing in the x-z plane.

Design cost cases are represented as a set of attribute-value pairs. This set is defined as a union of the design's function, behaviour and structure attributes. Function attributes relate to the design function, behaviour attributes relate to the design performance and structure attributes relate to the physical realisation of the design. The associated design cost model defines the abstract context of the design cost cases and may contain domain

knowledge that may be utilised for case retrieval, selection and adaptation. Figure 10 illustrates the structure of design cost models within the casebase.

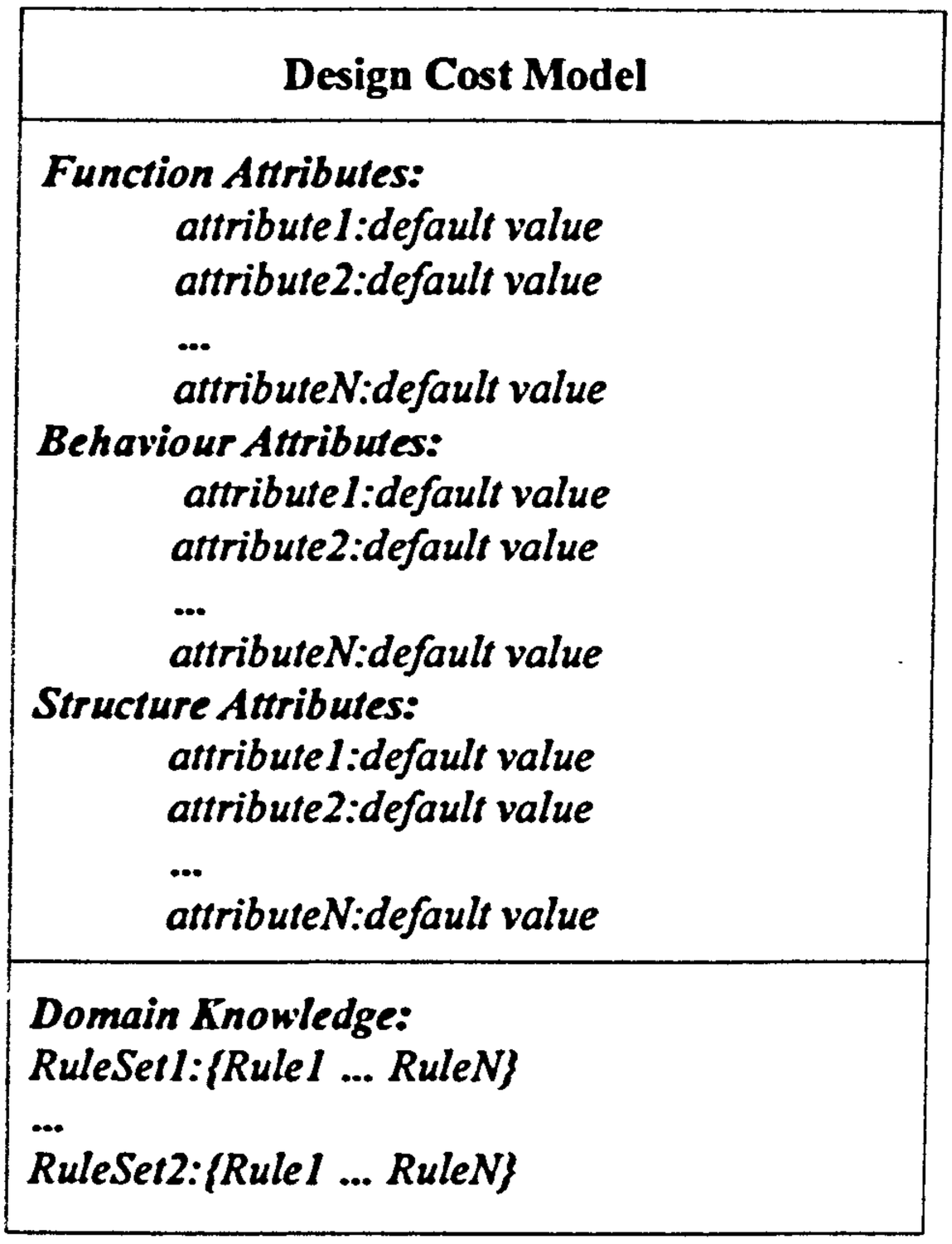


Figure 10. Design Cost Model Structure

4.3.2 Content of Design Cost Cases

The actual content of design cost cases is domain specific and can be determined through knowledge elicitation from a domain expert. One approach to deciding the content of a cost case is task analysis. In the task analysis carried out in Appendix A2.1, the expert is observed constructing cost estimates for components that are considered typical of their class. Consider Task Script 2 for the seaming turret component. This component can be said to belong to a class of components described as turrets. On analysis of the task script and examination of a number of designs that fall under the class of turret designs, one is able to identify the attributes that are generic to all such designs. This generic attribute set can be defined as the attributes of the turret design cost model. The elicited definition of the turret design model and the seaming turret case is illustrated in Figures 11 and 12, respectively.

Turret Design Cost Model
Function Attributes: *partDescriptor : string* ***Behaviour Attributes:*** *partCost : float* *materialCost: float* *manufacturingCost: float* *assemblyCost: float* *subContractCost: float* *makeorBuy : enumeration* *material : string* *isMaterialBoughtIn : Boolean* *materialProperties : list of strings* *assemblyHandlingCode: integer* *assemblyInsertionCode: integer* *lubricationMethod: enumeration* ***Structure Attributes:*** *manufacturingFeatureModel :* *list of manufacturing features*
Domain Knowledge: *Rule Set: Turret case adaptation.*

Figure 11. Turret Design Cost Model

4.3.2 Collecting Design Cost Cases

Case collection is an important issue in a domain like machine design in which if a case base is to be representative of the problem domain, it must potentially contain a large number of design cost models and their associated design cost cases. The collection of design cost models must be done manually, as they are unlikely to be specified explicitly in a company's databases. Therefore, design cost model specification is done traditionally by knowledge elicitation from domain experts.

Once the design cost models are specified they can act as templates for adding new design cost cases and consideration can be given to either automating the case collection process or making it interactive.

Turret Design Cost Case

Function Attributes:

partDescriptor :
(seaming_turret:main_turret_assembly:orbit_machine)

Behaviour Attributes:

partCost : 2322.62
materialCost : 911.56
manufacturingCost:846.64
assemblyCost:564.42
subContractCost:None
makeorBuy : make
material : STEEL EN 8
isMaterialBoughtIn : True
materialProperties : flame_cut o/diameter
flame_cut i/diameter
flame_cut bores
stress relieved
grind to flatness tol.
grind to parallelism tol.
lubricationMethod: oil
assemblyHandlingCode: 99
assemblyInsertionCode:39

Structure Attributes:

manufacturingFeatureModel:
this_turret's_feature_model

Figure 12. Seaming Turret Case

4.3.4 Recalling Design Cost Cases

The definition of an effective case recall protocol is of critical importance in a case-based system that operates in a domain where there may be a large number of cases as in machine design. The case recall protocol has to specify how cost cases are to be indexed, retrieved and finally selected. The representation and structuring of cases in the case base determines what indexing schemes may be employed. Therefore, much of the decision making that went into defining a representation scheme earlier was strongly influenced by the issue of case indexing for case recall.

The defined representation scheme enables 3-dimensional case browsing. Such a scheme lends itself readily to the use of both index trees and feature-based indexing. Here, case indexing is done by combining index trees and feature-based indexing. This

leads to a flexible and efficient indexing scheme. The case descriptors used in the index tree are simply the model/sub-model identifiers. These identifiers are the primary keys to the case. Cases may also be indexed by several of their constituent attribute-value pairs. These key-value pairs act as secondary keys to the case.

In line with the defined indexing scheme, case retrieval is done by a combination of concept refinement and associative recall. This enables the efficient retrieval of relevant cases that is those cases that are identified as being similar to the new case specification/definition. The definition of similarity can be relaxed or tightened according to the content of the case base. Here, the basic definition of similarity is a fairly relaxed one in which as long as cases of the same design component class as the new design component exist in the case base, they will be deemed to be similar. However, the designer is free to further refine this definition during design context specification and index elaboration.

Design context specification involves the setting of the primary case keys. Here, the primary keys are defined as the machine model descriptor, the assembly model descriptor and the part model descriptor. Any combination of these keys can be specified, however at least one primary key has to be specified in order to begin the case retrieval. Once a primary key is specified, index elaboration is enabled, in which secondary keys can be specified. Their specification can be in the form of a simple assignment or any other relational operation (i.e. $\leq$, $<$, $>$, $\geq$, !=, $\subset$). The specified key-expression pairs are added to the initial specification. The precedence of the secondary keys can also be specified as an indication to the case retrieval mechanism of the relative importance of the keys. The specified primary and secondary keys are used to construct an ordered set of case filters. The case filters are ordered as follows. Secondary case filters, the order of which is decided by the specified precedence order, if any, follow primary case filters. The case retrieval process involves the iterative filtering of the case base by application of this set of case filters to yield a set of most similar cases. The cases within this filtered set are considered to be on par with each other as far as their similarity to the new design cost case is concerned. Therefore, case selection can be at random from this filtered set.

The protocol for case retrieval is defined in the following rule set:

RULESET (Case Retrieval)
Execution Strategy: EACH (i.e. each rule in the set that evaluates to true is to be fired)

Rule 1:
IF (a machine model descriptor is specified)
THEN
Allow index elaboration for machine model;
Search casebase with specified indices;
Retrieve the set of matching machine cases;
AND IF (no matching cases found)
THEN
Report "No Match";

Rule 2:
IF (a assembly model descriptor is specified)
THEN
Allow index elaboration for assembly model;
AND IF (a set of matching machine cases exists)
THEN
Search this set with specified indices;
Retrieve the set of matching assembly cases;
AND IF (no matching assembly cases found)
THEN
Search casebase with specified keys;
Retrieve the set of matching assembly cases;
AND IF (no matching cases found)
THEN
Report "No Match";

Rule 3:
IF (a part model descriptor is specified)
THEN
Allow index elaboration for part model;
AND IF (a set of matching assembly cases exists)
THEN
Search this set with specified indices;
Retrieve the set of matching part cases;
AND IF (no matching part cases found in set) AND
(a set of matching machine cases exists)
THEN
Search this set with the specified keys;
Retrieve the set of matching part cases;
AND IF (no matching part cases found)
THEN
Search casebase with specified keys;
Retrieve the set of matching part cases;
AND IF (no matching cases found)
THEN
Report "No Match";

The protocol for index elaboration and search is defined in the following rule set:

RULESET (Index Elaboration and Search)
Execution Strategy: FIRST (i.e. the first rule in the rule set that evaluates to true is to be fired)

Rule 1:
IF (secondary keys are specified)
AND IF (some precedence order is specified)
THEN
BEGIN with a current precedence value = 1
FOR (each case in the current case set)
IF (case satisfies the constraint specified by key with current precedence value)
Add the case to a new subset of retrieved cases;
IF (a new subset of retrieved cases exists)
Set the current case set to the new subset;
Increment the current precedence value by 1;
END when precedence value is equal to the maximum specified precedence.
ELSE
Search the current case set with the elaborated indices;
Retrieve the set of matching cases;

4.3.5 Adapting Design Cost Cases

Case adaptation is done by a combination of substitution, transformation and derivation, depending upon the extent of domain knowledge that may be associated with a design cost model and the level of design detail available. Domain knowledge may be associated with each design cost model to measure the level of incompleteness of the design. For example, here in the Turret Design Cost Model, the level of incompleteness is defined by the following rule:

If (the number of stations attribute has a value for the new machine design)
Then

Level of design completion (%) = (the number of design features currently present in the new component design x the number of stations in the recalled machine design) / (the number of design features present in the recalled component design x the number of stations in the new machine design) x 100.

Examining the cost modelling strategy illustrated in Figure 8. Given a new design specification, the first step in the strategy is to retrieve the closest matching case from the design cost case base by following the defined protocol for case recall. Once a case is selected, it is made available globally in the cost modeller.

The actual design specifications are entered by the designer who is able to revise the design specifications at any time during the cost modelling session. The case recall protocol is invoked each time the specifications are revised. The designer is free to modify the design at any product level during a cost modelling session. The cost modeller detects any changes in the design and adapts the selected case according to the case adaptation protocol that may be associated with the design model that the case is an-instance-of. The main aim of case adaptation is to update the behaviour attributes related to *cost* for the new design specification. Therefore, design cost case adaptation must produce estimates for the following cost contributors:

- Material Costs,
- Assembly Costs,
- Manufacturing Costs,
- Sub-contract costs.

In the adaptation protocol, adaptation knowledge can be associated with any of the case attributes. For each design cost model, the adaptation strategy for each model attribute can be specified first through analysis, in order to assist in defining a case adaptation protocol. The definition of the case adaptation protocol then involves associating adaptation rules with each attribute. This is illustrated by example in what follows.

Consider for example the definition of an adaptation protocol for the turret design cost model. The adaptation strategy associated with each model attribute is as follows:

Turret Design Model Attributes:	Adaptation Strategy:
PartDescriptor:	Substitution
PartCost:	Derivation
MaterialCost:	Derivation
ManufacturingCost	Derivation
AssemblyCost:	Derivation
SubContractCost:	Derivation
MakeOrBuy:	Substitution
Material:	Substitution
IsMaterialBoughtIn:	Substitution
MaterialProperties:	Substitution
LubricationMethod:	Substitution
AssemblyHandlingCode:	Substitution and/or Transformation
AssemblyInsertionCode:	Substitution and/or Transformation
ManufacturingFeatureMmodel	Substitution and/or Transformation

The adaptation strategy associated with each attribute can be classified as substitution, derivation or a combination of substitution and transformation. If the adaptation strategy associated with an attribute is one of substitution then the new specified value simply replaces the old attribute-value.

If the strategy is one of derivation then the associated adaptation rule is one that utilises the same problem-solving process as that utilised in the selected case to arrive at an adapted attribute value. Derived attributes are unlikely to be assigned actual values in the design specification. They are more likely to be initially associated with some constraint for case recall purposes.

If the strategy is one of a combination of substitution and transformation then the associated adaptation rule evaluates the new specification before deciding which adaptation strategy to follow. If the evaluation concludes that transformation is the approach to take then adaptation occurs by the application of domain knowledge specifically devised for adapting that particular attribute. For example, among the structure attributes of a design model is the manufacturing feature model associated with the design case. One of the functions of the adaptation process is to consider the feature model of the selected design case along with the incomplete feature model (if one exists) of the new design and merge the two in order to produce an adapted feature model. The aim of adaptation here is to patch up the incomplete description of the new design for the purpose of cost estimation.

Consider the feature model for the seaming turret component shown in Figure 13. This feature model is based on the feature model template used in the case study (Figure 14). The adaptation strategy associated with the *manufacturingFeatureModel* attribute is one of a combination of substitution and transformation. This strategy is taken in order to allow costs to be modelled throughout the design phase, from conceptual to detail design.

For example, if no actual design has been done, but the number of stations and the part model in the new machine has been specified, then the feature model of the selected case can be adapted to meet the new specification. This is done by applying parametric rules that may be defined for the design model. As the designer has not altered the feature model in the specification or begun actual design, the adaptation at this stage will not produce an innovative design description. Therefore, at this level one can say that the trade-off between case-based design completion and innovation is at a

maximum. However, as soon as design is entered into this trade-off decreases in line with the level of design detail available.

If actual design is detected to have taken place, then the adaptation strategy is to firstly apply feature recognition knowledge in order to obtain an incomplete design description in the form of a manufacturing feature model. This feature model is then transformed by applying the associated captured domain knowledge.

The protocol for adapting a selected design cost case to meet a new design's specification may be defined by the following rule set:

RULESET (Turret Case Adaptation)
Execution Strategy: EACH (i.e. each rule antecedent is tested and those that evaluate to true are fired)

Rule 1:
IF(A design specification exists)
AND IF (a closest matching case has been selected from the case base)
AND IF (no design has taken place)
THEN
Invoke the Substitution Task;
Invoke the Parametric Adaptation Task;
IF (an adapted case exists)
Invoke the Cost Estimation Task;
ELSE
Invoke the Substitution Task;
Invoke the Feature Recognition Task;
IF (a partially complete feature model for the new design exists)
THEN
Invoke the Transform Feature Model Task;
IF (a partially complete feature model for the new design exists)
Invoke the Cost Estimation Task;
END RULE

The Feature Recognition and the Cost Estimation tasks mentioned in the adaptation protocol are discussed in the sections that follow on Design Interpretation and the Capture and Application of Cost Knowledge.

PAGE NUMBERING AS ORIGINAL

4.4 Design Interpretation

Analysis of the elicited task scripts (Appendix A2.1) in the case study reveals that the expert interprets the design as a set of manufacturing features to which he then applies the machining rules. In order to mimic the expert the cost modeller has to address the design interpretation issue namely that of eliciting a manufacturing feature model of the design from CAD. The two main approaches to creating feature models for designs are that of feature recognition and design-by-features. The former interrogates geometric models of the CAD design in order to construct a feature model representation while the latter constructs the CAD design by using pre-defined features and creates the geometric models from this feature-based representation. While the branch of feature recognition is the most mature it is yet to be widely commercially available while on the other hand CAD systems incorporating the Design-By-Features approach are already commercially available (e.g. Pro-Engineer and I-DEAS) and their functionality is rapidly increasing.

The approach to design interpretation taken here, however, is that of feature recognition. The choice of this approach is mainly driven by observation of the designer's practice within the current case study at CMB Engineering. The designers at CMB use a state-of-the-art CAD system, I-DEAS, which incorporates both the traditional design creation techniques as well as design-by-features. It was observed that the majority of the time, the designers chose to use the traditional design creation techniques because they found it be more intuitive and less restrictive than the design-by-features approach. In order to allow the designers to continue their design practice as normal, automatic feature recognition was seen as the best approach to design interpretation in this study. A literature survey of the different approaches to feature recognition is given in section 3.4.

The choice of method used for feature recognition is driven by the geometric representation used by the CAD system. Feature recognition is complicated and no one method has emerged that outperforms the others. Here, a hybrid approach to feature recognition is taken. This is a feasible approach because the emerging standard in state-of-the-art CAD systems is to incorporate all the different geometric models in order to

achieve an optimised CAD facility. All of the following geometric models may be utilised each bringing with it a particular benefit:

- CSG model representation insures that only valid objects can be created;
- Boundary model representation yields good graphics for high quality surface representations;
- Wireframe model representation allows rapid visualisation and geometry definition; and,
- Sweep model representation provides easy-to-use methods for geometry construction.

The computational overhead incurred in providing all these geometric models is not significant. This is because the boundary model can be readily derived from the sweep model which itself can act as a fundamental building block of the CSG model.

The feature recognition algorithm defined here falls under the category of hybrid, rule-based automatic feature recognition. The prescribed algorithm utilises the CSG model along with its associated sweep and wireframe representations. The process model for the defined feature recognition facility is illustrated in Figure 15. In CSG models, the geometry is represented as a binary tree of Boolean operations that are usually applied to a limited set of fundamental shape primitives. However, the combination of wireframe and sweep representations as building blocks of CSG models provide a rich geometric modelling facility in which the range of primitives is vastly increased without predefinition.

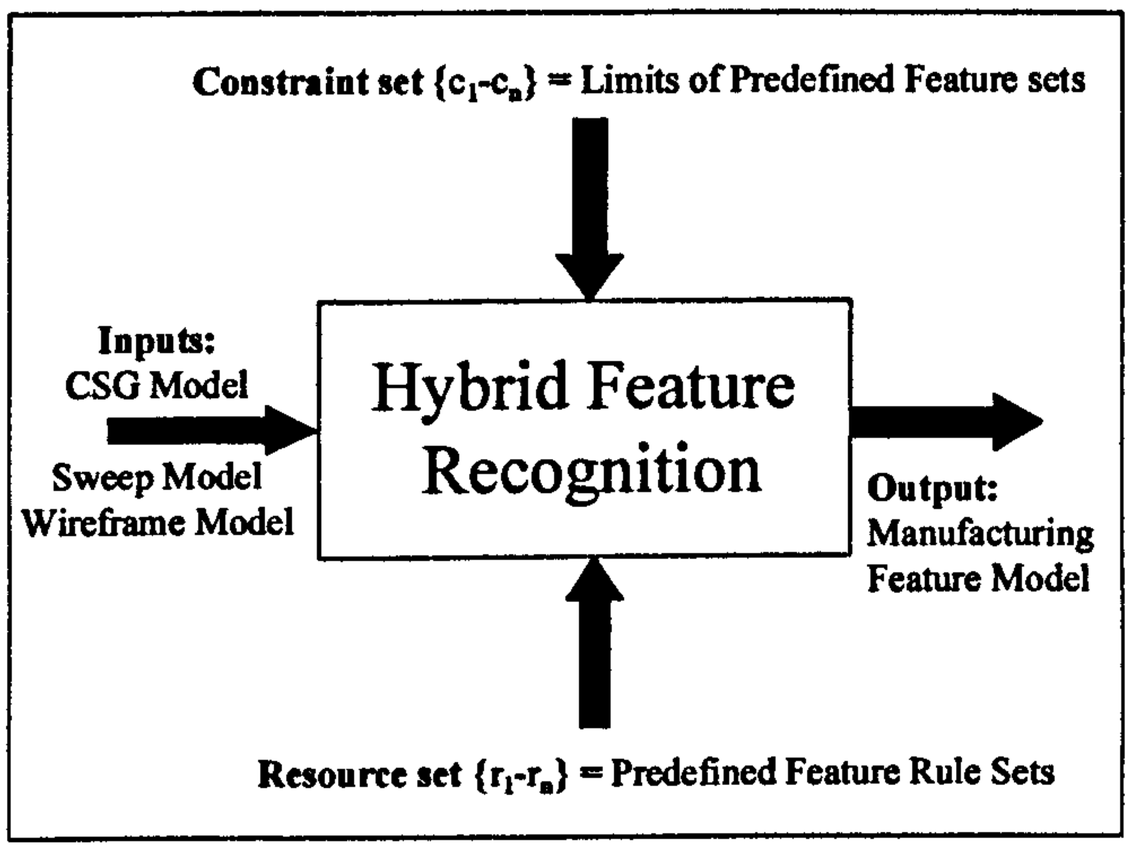


Figure 15. Feature Recognition Process model

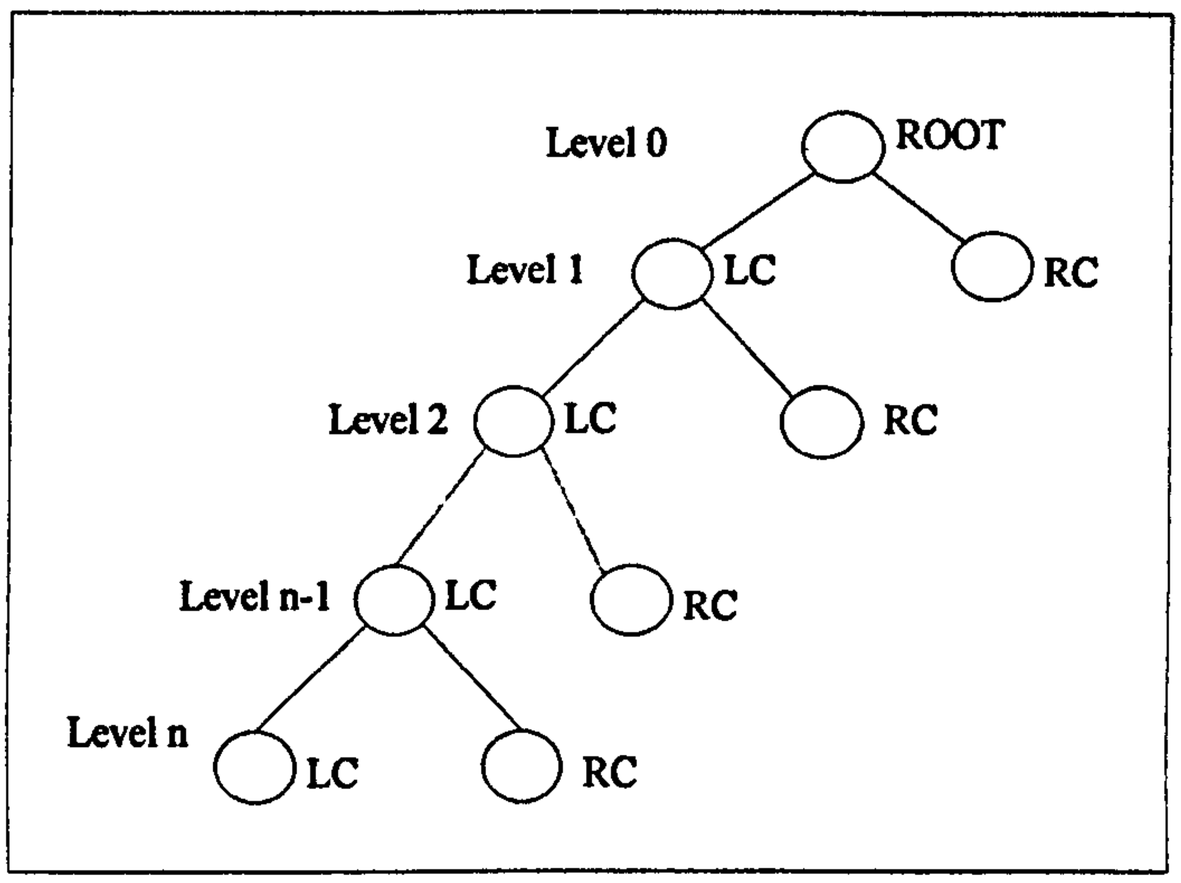


Figure 16. CSG tree structure

Consider, the generic CSG tree shown in Figure 16. The leaves of the tree represent primitives that can either be predefined or constructed by applying a sweep operation to a wireframe definition. The intermediate nodes represent the Boolean operation applied and store the resulting Geometry State.

The feature recognition algorithm applied here is defined by the rule set on page 59. In this rule set, the actual features are recognised by searching the associated wireframe section of a sweep operation for pre-defined 2-D features. *Peters(1991)* presents a method for 2-D feature recognition that maps the recognised features to the associated 3-D manufacturing features by utilising a neural net. Here, 2-D features are recognised and then mapped to the associated set of 3-D features by the application of feature recognition rules. These rules infer the design context in which the 2-D features were created by analysing the details of the associated sweep operation.

Here, the defined hybrid method for feature recognition firstly reduces the search space significantly by only considering a closed wireframe section when searching for features, and secondly, uses knowledge to guide the feature recognition. Therefore, this approach to rule-based feature recognition eliminates the computationally expensive repeated blind search of the solid model, usually associated as the greatest drawback of rule-based feature recognition. A rule-based approach to feature recognition is particularly attractive when the features of interest are domain-dependant, as is the case here where we are interested in recognising machining features.

```
RULESET (Automatic Feature Recognition)
Execution Strategy: EACH

IF (a CSG (history tree) exists for the component)
   THEN
        From the root node of the CSG tree at level 0;
        Traverse down the left child branch of the tree to tree level n-1;
        Query the Boolean operator applied to decide whether the LC or RC of
        The node at level n-1 represents the initial feature node in the part history tree;
        Traverse down to the elicited initial feature node;
        IF (initial feature node is a Constructed Primitive)
                AND IF (the construction operation is sweep (extrude))
                    THEN
                        Search for pre-defined prismatic features in the associated wireframe section;
                ELSE IF (the construction operation is sweep (rotate))
                     THEN
                        Search for pre-defined rotational features in the associated wireframe section;
        ELSE IF (initial feature node is a Pre-Defined primitive)
                THEN
                        Retrieve the pre-defined features;
        END IF
Output feature model outlining the billet shape and the initially defined manufacturing features;
Until (n = 0)
        Traverse up the tree to the parent feature node;
        Traverse down to the child leaf node not yet visited;
        Search for pre-defined features in this node's associated wireframe section;
        IF (the construction operation is a cut (difference operator))
          THEN
                Mark the pre-defined features as negative volumes (cutting features);
        IF (the construction operation is a protrude (union operator))
          THEN
                Mark the pre-defined features as positive volumes (protruding features);
        Update Feature Model.
```

Consider for example the spring-plunger component design in Appendix A4. One history tree (designs do not have unique history trees) for the construction of this design in a solid modeller is as illustrated in Figure 17. Here, the solid model is constructed by first rotating a closed wireframe section through 360°. As the leaf node operation for the primary creation operation of the part is one of rotation, the part can be inferred to be rotational. The wireframe section associated with this sweep operation contains all the internal and external features defined in the axial direction of this rotational part. That is the following feature set: {4 grooves, 2 outside/diameters, 4 chamfers, two undercuts, a recess hole, a counter-bore}. In a defined 2-D feature set, one can imagine how one 2-D feature may map to a number of 3-D features depending upon the context of the wireframe definition and the associated sweep operation. Therefore, rules to determine the 2-D feature context must be defined.

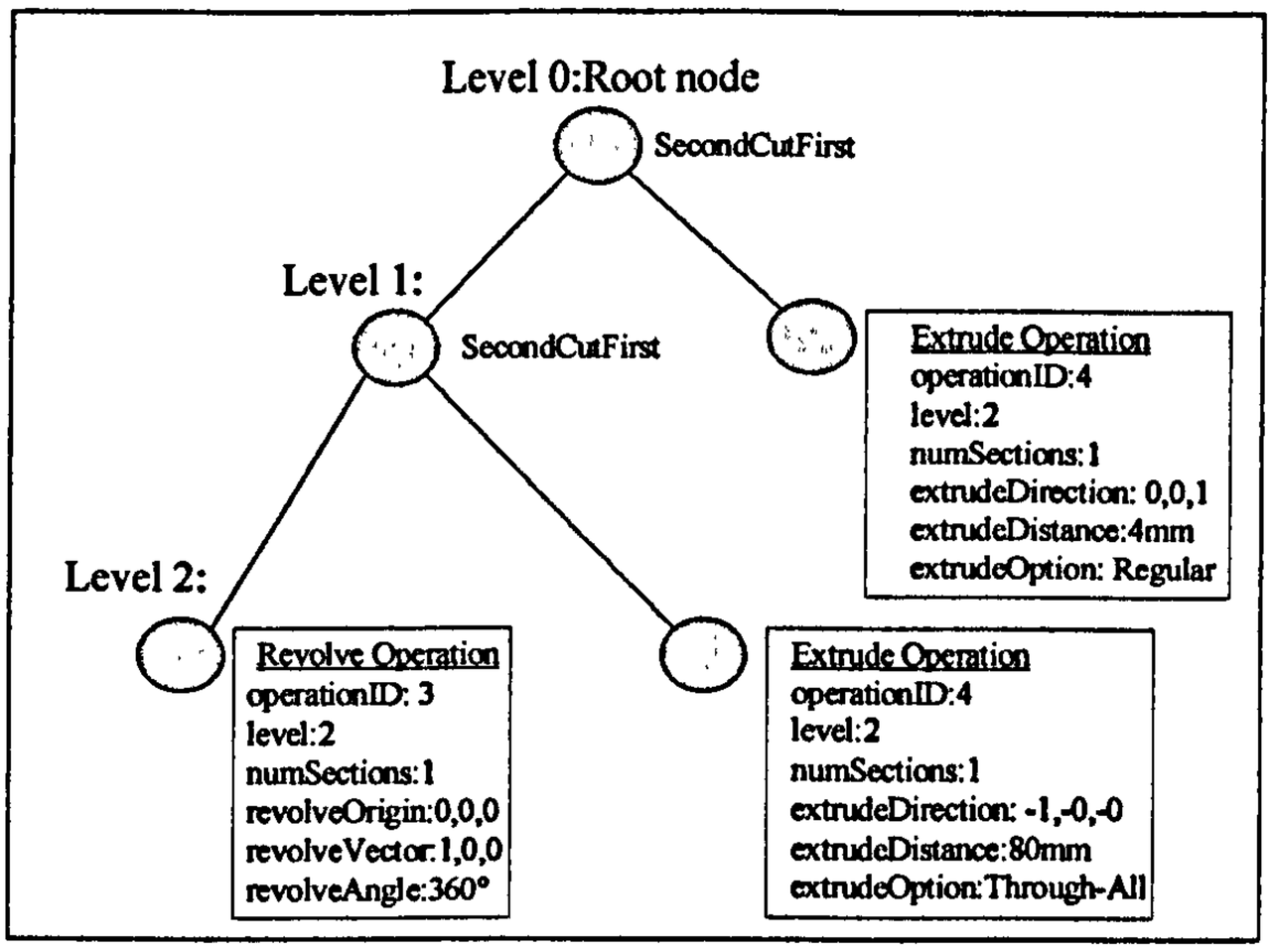


Figure 17. A History tree for Spring-Plunger Component Design

The first step taken at each tree node by the feature recognition algorithm is to operate on the wireframe section associated with that design node. A wireframe section is a closed profile consisting of connecting wireframe curves. It is naturally organised by its definition as a vector of curves. The wireframe curve elements in this vector are described here as curve objects containing the following attribute set to enable query: {*wf-curve-form-type*: enum; used here to explicitly specify the analytic curve intent; *wf-curve-start-point* data and *wf-curve-end-point* data}. The curveFormType may be specified as either one of the following set: {non-rational-spline, line-segment, circular_arc, elliptic_arc, parabolic_arc, hyperbolic_arc, composite, rational-spline or a degenerate.

The wireframe section associated with the initial spring-plunger part creation feature node is represented by the vector of connected wireframe curves on page 61. This vector places the wireframe curves in their clockwise order of occurrence. The curve at the front of the vector holds no particular significance. The wireframe section is a closed section so the curve at the front of the vector is connected to the curve at the back of the vector. The feature recognition proceeds by searching the wireframe vector for pre-defined 2-D features. Each 2-D feature found is mapped to the associated 3-D machining feature by checking the sweep operator being applied. For example, the four grooves in the spring-plunger are axial features along the cylindrical element with an outside diameter of 54mm. In the wireframe each groove is defined by the pattern: line-

segment1:circular-segment:line-segment2, where line-segments1 and 2 have the same height. This pattern is searched for in the vector. If found the associated sweep operation is queried. If a rotation operation is inferred and the groove pattern is detected along the diameter of a cylindrical element, it is inferred to be a cylindrical groove. If the pattern is detected in the face of a cylindrical element, it is inferred to be a circular facial groove. If the sweep operation is one of extrusion, then the pattern is detected as a facial groove in the direction of extrusion. Such a context query process is applied to each detected 2D-feature until no more feature matches are found. A rule set for eliciting the manufacturing feature model from this vector and its associated sweep operation is defined overleaf.

Wireframe Section Vector

index (i)	wf-start_point	wf_end_point	wf_curve_form_type
0	36,27,0	40,27,0	Line-segment
1	40,27,0	40,25,0	Line-segment
2	40,25,0	41,24,0	Circular-arc
3	41,24,0	42,24,0	Line-segment
4	42,24,0	43,25,0	Line-segment
5	43,25,0	78,25,0	Line-segment
6	78,25,0	80,23,0	Line-segment
7	80,14,0	80,23,0	Line-segment
8	79,13,0	80,14,0	Line-segment
9	79,13,0	24,13,0	Line-segment
10	24,13,0	23,14,0	Line-segment
11	23,21,0	23,14,0	Line-segment
12	2,21,0	23,21,0	Line-segment
13	2,21,0	0,23,0	Line-segment
14	0,23,0	0,27,0	Line-segment
15	0,27,0	32,27,0	Line-segment
16	3.2,27,0	67,27,0	Circular-arc
17	6.7,27,0	13.2,27,0	Line-segment
18	13.2,27,0	16.7,27,0	Circular-arc
19	16.7,27,0	23.2,27,0	Line-segment
20	23.2,27,0	26.7,27,0	Circular-arc
21	26.7,27,0	33.2,27,0	Line-segment
22	33.2,27,0	36.7,26.9,0	Circular-arc

What follows are excerpts from the rulesets for manufacturing feature model construction and context based feature extraction defined here. These rulesets illustrate the reasoning behind the feature recognition approach taken.

RULESET (Manufacturing Feature Model Construction)
Inputs: Design Feature Node and Associated Wireframe Section Vector
Execution Strategy: Each

IF (the *x* co-ordinate remains constant for all wireframe connector points)
THEN
    Wireframe is defined in the Y-Z plane;
ELSE IF (the *y* co-ordinate remains constant for all wireframe connector points)
THEN
    Wireframe is defined in the X-Z plane;
ELSE IF (the *z* co-ordinate remains constant for all wireframe connector points)
THEN
    Wireframe is defined in the X-Y plane;
ELSE
    Report 'Wireframe definitions in planes other than the principle X-Y, X-Z and Y-Z planes are
        not handled in this ruleset definition';

IF (initial part creation operation is a REVOLVE)
    Create a *MainlyTurned, DesignComponent* Instance;
    IF (the REVOLVE vector is 1,0,0 and the wireframe is defined in the X-Y Plane)
    THEN
     Set the *BoundingCylinderLength* to the difference between the maximum and minimum *x*- co-ordinates occurring in the wf-vector;
     Set the *BoundingCylinderDiameter* to twice the difference between the *y*- co-ordinate of the revolve origin and the maximum *y*- co-ordinate occurring in the wf-vector;
     BEGIN with a wireframe curve whose wf-start-point *y*- co-ordinate is equal to the maximum *y*- co-ordinate occurring in the wireframe curve vector;
     For (each wireframe curve, $curve_i$, in the wireframe curve vector)
        Apply the defined context-based feature extraction ruleset;
ELSE IF (the REVOLVE vector is 0,1,0 and the wireframe is defined in the X-Y Plane)
THEN
     Set the *BoundingCylinderLength* to the difference between the maximum and minimum *y*- co-ordinates occurring in the wf-vector;
     Set the *BoundingCylinderDiameter* to twice the difference between the *x*- co-ordinate of the revolve origin and the maximum *x*- co-ordinate occurring in the wf-vector;
     BEGIN with a wireframe curve whose wf-start-point *x*- co-ordinate is equal to the maximum *x*- co-ordinate occurring in the wireframe curve vector;
     For (each wireframe curve, $curve_i$, in the wireframe curve vector)
        Apply the defined context-based feature extraction ruleset;
ELSE
    Report 'Wireframe sections revolved about axis other than the principle planar X and Y axis are not handled in this ruleset definition';

IF (the REVOLVE vector is 1,0,0 and the wireframe is defined in the X-Z Plane)
THEN
     Set the *BoundingCylinderLength* to the difference between the maximum and minimum *x*- co-ordinates occurring in the wf-vector;
     Set the *BoundingCylinderDiameter* to twice the difference between the *z*- co-ordinate of the revolve origin and the maximum *z*- co-ordinate occurring in the wf-vector;
     BEGIN with a wireframe curve whose wf-start-point *z*- co-ordinate is equal to the maximum *z*- co-ordinate occurring in the wireframe curve vector;

For (each wireframe curve, $curve_i$, in the wireframe curve vector)
Apply the defined context-based feature extraction ruleset;
ELSE IF (the REVOLVE vector is 0,0,1 and the wireframe is defined in the X-Z Plane)
THEN
Set the *BoundingCylinderLength* to the difference between the maximum and minimum *z*- co-ordinates occurring in the wf-vector;
Set the *BoundingCylinderDiameter* to twice the difference between the *x*- co-ordinate of the revolve origin and the maximum *x*- co-ordinate occurring in the wf-vector;
BEGIN with a wireframe curve whose wf-start-point *x*- co-ordinate is equal to the maximum *x*- co-ordinate occurring in the wireframe curve vector;
For (each wireframe curve, $curve_i$, in the wireframe curve vector)
Apply the defined context-based feature extraction ruleset;
ELSE
Report 'Wireframe sections revolved about axis other than the principle planar X and Z axis are not handled in this ruleset definition';

IF (the REVOLVE vector is 0,0,1 and the wireframe is defined in the Y-Z Plane)
THEN
Set the *BoundingCylinderLength* to the difference between the maximum and minimum *z*- co-ordinates occurring in the wf-vector;
Set the *BoundingCylinderDiameter* to twice the difference between the *y*- co-ordinate of the revolve origin and the maximum *y*- co-ordinate occurring in the wf-vector;
BEGIN with a wireframe curve whose wf-start-point *y*- co-ordinate is equal to the maximum *y*- co-ordinate occurring in the wireframe curve vector;
For (each wireframe curve, $curve_i$, in the wireframe curve vector)
Apply the defined context-based feature extraction ruleset;
ELSE IF (the REVOLVE vector is 0,1,0 and the wireframe is defined in the Y-Z Plane)
THEN
Set the *BoundingCylinderLength* to the difference between the maximum and minimum *y*- co-ordinates occurring in the wf-vector;
Set the *BoundingCylinderDiameter* to twice the difference between the *z*- co-ordinate of the revolve origin and the maximum *z*- co-ordinate occurring in the wf-vector;
BEGIN with a wireframe curve whose wf-start-point *z*- co-ordinate is equal to the maximum *z*- co-ordinate occurring in the wireframe curve vector;
For (each wireframe curve, $curve_i$, in the wireframe curve vector)
Apply the defined context-based feature extraction ruleset;
ELSE
Report 'Wireframe sections revolved about axis other than the principle planar Y and Z axis are not handled in this ruleset definition';

...

RULESET (Context-Based Feature Extraction)
Inputs: $curve_i$, $curve_{i-1}$, $curve_{i+1}$
Execution Strategy: First

IF (the wireframe section is defined in the XY plane)
IF ($curve_i$ is a straight line segment in the axial direction and
the *x*- co-ordinate of wf-start-point of $curve_i$ is less than that of the wf-end-point)
THEN
This curve represents a *CylindricalElement* Manufacturing Feature of diameter $2(y-y_0)$;
IF (a *CylindricalElement* of diameter 2y is not already part of the Feature Model)
Add a *CylindricalElement* Manufacturing Feature to the Feature Model;

IF ($curve_i$ is a straight line segment in the axial direction and
the *x*- co-ordinate of wf-start-point of $curve_i$ is greater than that of the wf-end-point)
THEN
This curve represents a *Hole* Manufacturing Feature of diameter $2(y-y_0)$ and length $|(x_2-x_1)|$;

IF (*curve*$_i$ is a Circular-Arc and
curve$_{i-1}$ is a straight Line-Segment in the axial direction and
curve$_{i+1}$ is a straight Line-Segment in the axial direction and
the *y*- co-ordinates of wf-start-point and wf-end-point of *curve*$_i$ are equal and
the *x*- co-ordinate of wf-start-point of *curve*$_i$ is less than that of the wf-end-point)
THEN
Add a *CylindricalGroove* Manufacturing Feature to the Feature Model;

IF (*curve*$_i$ is a Circular-Arc and
curve$_{i-1}$ is a straight Line-Segment perpendicular to the axial direction and
curve$_{i+1}$ is a straight Line-Segment perpendicular to the axial direction and
the *x*- co-ordinates of wf-start-point and wf-end-point of *curve*$_i$ are equal and
the *y*- co-ordinate of wf-start-point of *curve*$_i$ is greater than that of the wf-end-point)
THEN
Add a *FacialGroove* Manufacturing Feature to the Feature Model;

IF (*curve*$_i$ is a Circular-Arc and
curve$_{i-1}$ is a straight Line-Segment and
curve$_{i+1}$ is a straight Line-Segment and
the *y*- co-ordinates of wf-start-point and wf-end-point of *curve*$_i$ are unequal and
the *x*- co-ordinate of wf-start-point of *curve*$_i$ is less than that of the wf-end-point)
THEN
Add a *UnderCut* Manufacturing Feature to the Feature Model;

...

4.5 Capture and Application of Cost Heuristics

The success of modelling costs using a knowledge-based approach is greatly dependent on the quality and completeness of domain knowledge. Therefore, knowledge elicitation is an important issue that needs to be addressed. Many techniques exist to assist in the knowledge acquisition process and the reader is referred to *Gonzalez and Dankel(1993)* who discuss the issue of knowledge acquisition in greater depth.

Although careful consideration has to be given to the capture and application of cost heuristics, the treatment of this issue is situation-dependent and intuitive. Ultimately, the knowledge engineer has to decide which approach to follow, taking into account the resources available and any constraints on the process, such as gaining access to suitable experts. In the case study considered here the approach to knowledge elicitation taken is one-to-one interviews with domain experts. In the absence of suitable experts, published literature on the domain is used. Elicited knowledge that is directly relevant to this report is contained in Appendix-A.

Once the knowledge has been elicited, it has to be structured in a manner appropriate for reasoning about the problem domain. The main paradigm adapted throughout this research work has been one of object-orientation. Therefore, domain knowledge is structured by an object-oriented analysis. The object model for the cost modelling problem domain is illustrated in Chapter 6 on prototype development.

Another paradigm adapted in this research is a feature-based one. This is because much of the domain expert's reasoning about the problem domain is feature-based. Examples of research work reported in the literature on features and manufacturing cost analysis include *Nieminen and Toumi(1991), Opas and Mantyla(1994)* and *Wozny et al.(1994).*

4.6 Chapter Summary

This chapter has outlined the designed methodology for knowledge-based cost modelling for innovative design. The placement of enabling technology within the methodology is also justified here. The specification of the communication protocol at the heart of the designed cost modeller infrastructure is deferred to the next chapter on the system architecture.

---

5 System Architecture

5.1 System Requirements

The basic objective of the methodology is to establish a uniform, domain-independent approach to cost estimation for innovative design. The primary requirement is for cost estimates to be sufficiently accurate throughout the design phase. Therefore, at concept design, cost estimates produced must give a clear indication of the real scale of the production cost and as the design is detailed, cost estimates must be accurate enough to guide the designer to design to cost. A successful cost estimation method also has to reflect the user's requirements. From the designer's perspective, the cost estimation method is required to have the following characteristics:

- It must reflect the dynamic nature of the design process. Therefore, at any stage during the design phase, from concept to detail design, the designer should be able to invoke the system and receive a cost estimate.
- It must provide a good CAD interface in order to avoid prompting the designer excessively for a design description.
- It must be efficient in time in its delivery to the user. Therefore, the system should not have to go through the whole estimating process every time a design change is made. It should be able to detect any design changes and update the estimate accordingly.
- It must make the reasoning behind the cost estimate explicit to the user in order for the produced cost estimate to be used with confidence.
- It must provide a mechanism for linking with databases that may store information relevant to the cost estimation process (e.g. material costs, sub-contract operation costs and stock lists).
- It must have an easily extendible knowledge base. Therefore, the issues of modularity and reusability should also be considered in the design of the methodology.

These secondary requirements are really system requirements and can be met by taking the right architectural strategy. The strategy adopted is based on an advanced AI architecture, the blackboard, which has been successfully applied to a diversity of problem domains. The selection of this architecture is driven not only by the system requirements outlined above, but also by the need for an effective communication protocol that lies at the heart of the cost modeller.

5.2 Cost Modeller Communication Protocol

Reiterating, the expert's cost estimation process involves completing incomplete design descriptions and constructing rough process plans for the designs. There is a lack of well-defined criteria for determining when a sufficiently accurate design description has been synthesised or a sufficiently accurate process plan has been constructed. Therefore, from an automation perspective, the cost estimation process has poorly defined goals. In addition, the expert's reasoning in the construction of the cost estimate is determined by the evolving product design. Therefore, in automating the cost estimation task, an evolutionary solution process that does not require a predetermined reasoning path is needed. The blackboard framework of problem solving is chosen here because it incorporates such an incremental and opportunistic problem-solving approach. Opportunistic reasoning is also greatly favoured for applications where the problem solving knowledge can be decomposed into autonomous parts as is the case in this instance. The disadvantages usually associated with blackboard systems are that they require to be tailored to a particular application and that the actual decomposition of the problem domain can be difficult to determine. These issues are not of concern here because we are not interested in the development of a generic shell-like system and the cost modelling method described in Chapter 4 decomposes the problem domain into its component parts. What follows is a description of the blackboard architecture as applied here.

An analogy often used to describe the blackboard problem solving approach is that of a group of experts standing around a blackboard trying to solve a complex problem. The state of the solution is written on the blackboard and is visible to the experts. When an expert has something to contribute to the evolving solution, he indicates his intentions to

the chairman of the group who is responsible for controlling the problem solving process. The chairman monitors the experts and selects their contributions in order, according to a specific agenda. The chosen expert is allowed to either write his contribution onto the blackboard or dispute a fact already existing on the blackboard and change the solution path. This process continues until the experts have nothing else to add to the blackboard. On completion of the problem solving process, the blackboard data structure stores the state of the final solution. It may also store any failed solution paths. The storage of the failed solution paths is a useful aspect since it allows the designer to keep a history of the design options and incomplete design ideas. This allows the designer to backtrack more easily if a problem is encountered on the current design path. A schematic of the system architecture is shown in Figure 18.

An object-oriented perspective is taken on the blackboard architecture, which consists of a blackboard data structure, knowledge sources and a controller. The role of the blackboard data structure follows in line with the blackboard in the analogy. The data structure is responsible for storing the evolving solution and making it visible to the relevant experts. All communication between the experts is done through the blackboard in order to produce consistent solutions. Ideally, the blackboard data structure, in this application would have both a vertical and a horizontal dimension. The horizontal dimension would be employed to represent partial solutions that may overlap. A horizontal dimension is required in order to reflect the dynamic nature of the design process. It would portray temporal groupings of partial solutions; i.e. every time a design change is made the updated solution would be stored in parallel to the previous solution. The different solution states would be stored in the form of deltas, that is the difference between the intermediate solution states. Storing deltas is a more efficient approach than storing entire solution states both from the perspective of memory usage and computational processing.

This interpretation of the blackboard data structure allows a decision history to be kept. It also assists in making the system efficient in its delivery to the user. This is because the system is able to detect any design changes made, through comparison with the previous partial solution, and update the solution accordingly.

The vertical dimension would be employed to represent different levels of abstraction in the solution space. Currently, in the Cost Expert Prototype described in Chapter 6, only the vertical dimension is implemented.

The blackboard is a hierarchical approach to problem solving. It is therefore useful as a means of handling complexity as it allows a complex problem domain to be viewed at different levels of abstraction, making it easier to handle. In order to fit into such a hierarchy, the problem domain has to be appropriately decomposed. The problem domain is decomposed through the application of an object-oriented analysis as described in Chapter 6.

The role of the knowledge sources is to simulate the way the experts deploy their knowledge in the given analogy. It is the knowledge sources, in the blackboard framework, that are responsible for moving the state of the blackboard towards a solution state. They do this by responding to events that occur on the board. The various knowledge sources are independent of each other and effect each other indirectly by their contributions to the blackboard. The knowledge about the problem domain is decomposed into objects that can be placed on the blackboard. Each object is associated with a knowledge source that is responsible for knowing how to invoke the object methods. Knowledge sources are expressed in the form of situation-action rules. Each rule consists of:

- A trigger, which determines when the knowledge source becomes active;
- A precondition, which determines when a rule is executable; and,
- An action, which updates the state of the blackboard by invoking the associated object methods.

The role of the controller is to monitor the state of the blackboard and on the occurrence of an event co-ordinate the actions of the knowledge sources. The control mechanism employed in these architectures has to reflect an opportunistic problem solving approach. Hence, the selection of what to do next, at any stage of the problem solving process, is made while the problem is being solved. This is achieved by a knowledge guided control strategy. Such knowledge is also expressed in the form of knowledge

sources. Control knowledge sources are responsible for dictating the relative priority of domain knowledge sources. The control strategy specified by *Craig(1991)* can be summarised by the following rule set:

1. A blackboard event occurs on abstraction level n
2. The knowledge sources associated with level n are searched and their triggers evaluated against the event.
3. For knowledge sources whose triggers evaluate to true, knowledge source activation records are created, and stored in a data structure.
4. The preconditions of the knowledge sources activated are evaluated.
5. The executable knowledge source activation records are placed in a separate data structure.
6. This data structure is sorted in order of priority, by application of the control knowledge sources.
7. The KSAR given the highest priority is executed. This causes further blackboard events.
8. One of the control knowledge sources determines whether a solution has been reached. If it has then the process terminates; otherwise, the control loop is executed again.

Figure 18 illustrates the abstraction hierarchy used for automating the cost estimation process. When a part cost estimate is required, the part design description obtained from the CAD interface, is input at the lowest abstraction level for inspection by the design interpretation KSs. These knowledge sources generate the part feature model on the next abstraction level. Manufacturing cost KSs are then activated to construct the part cost estimate. Similarly, when an assembly cost estimate is required, the part to part connectivity information is obtained from the CAD interface and the assembly cost KSs activated. Assembly to assembly information is used to construct the product cost estimates. The case-based retrieval KSs can be activated by events on all levels. For example, events occurring on level one may activate the case-based retrieval KSs to retrieve similar part designs from the case base while events occurring on level three may activate the retrieval of complete product designs.

The role of the link with a relational database is to demonstrate how knowledge outside the system that is relevant to the cost estimation process can be accessed by the cost modeller.

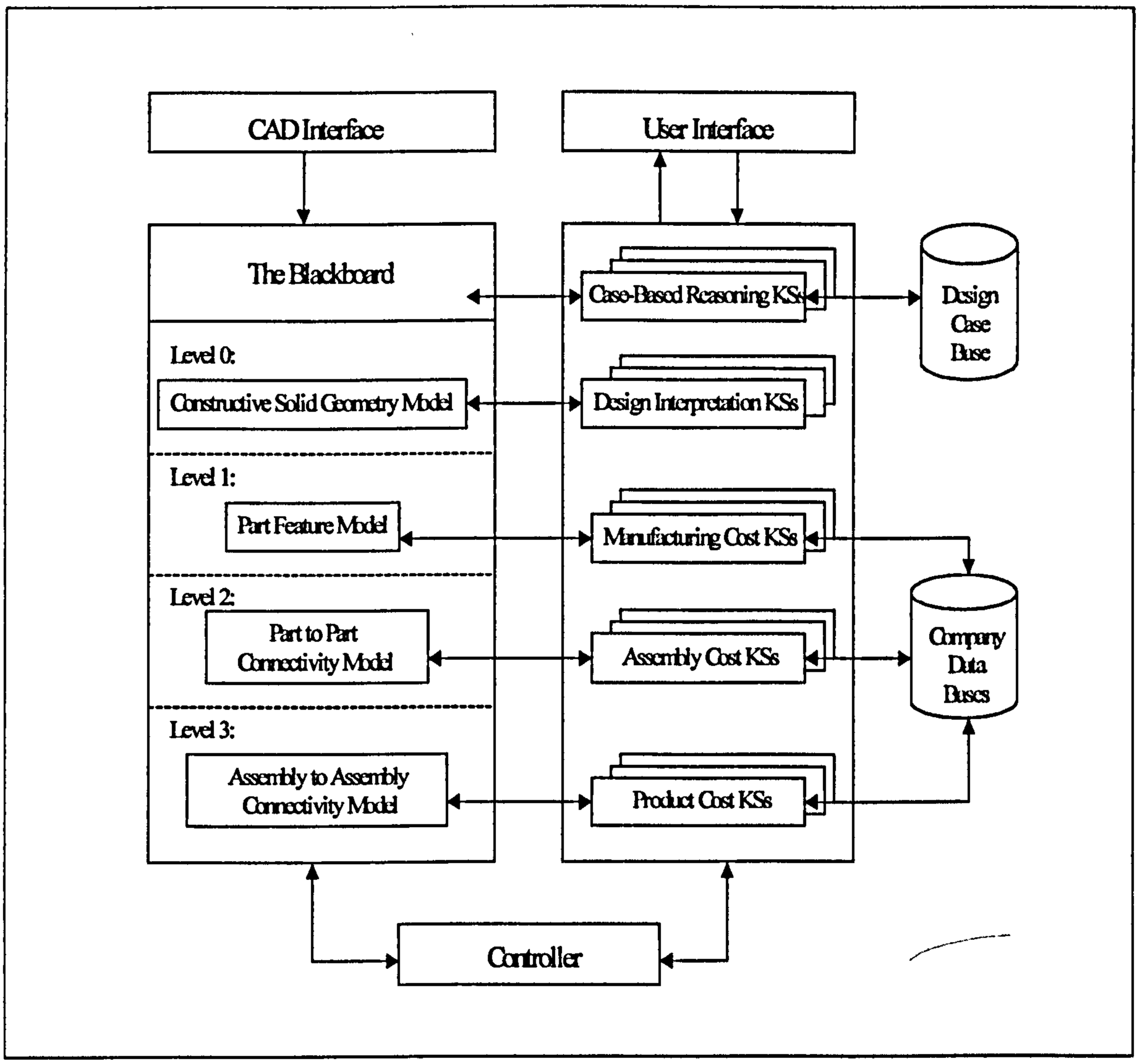


Figure 18. System Architecture

The role of the explanatory interface is to make the reasoning behind the cost estimate transparent to the user. This allows the user to gain confidence in using the produced cost estimate and assists the knowledge engineer in consistency checking.

6 Prototype Development

6.1 Introduction

Chapter 4 outlined a generic methodology for modelling costs throughout the design phase from conceptual to detail design aimed at innovative design activity. This methodology is treated as the system specification for the cost expert prototype. The cost expert prototype is described in what follows.

6.2 Prototype Development Environment

It was decided fairly early on in this research study that the prototyping approach taken would be an evolutionary one. This decision was mainly driven by the sponsor of the research, who wanted to see the methodology being demonstrated in the development environment envisaged for the final system i.e. one that was compatible with the sponsoring company's working environment.

The prototyping environment was therefore chosen to be within the UNIX operating system on SUN workstations, in the C++ programming language, with an X-Motif user interface and linked to the state-of-the-art I-DEAS CAD system.

6.2 System Design Strategy

An evolutionary prototype has to cater for future changing requirements. The system requirements were therefore specified to be as those defined in Chapter 5. Chapter 5 outlines an architectural strategy that accommodates these requirements.

The system design strategy taken is an object-oriented one. The requirement for extensibility is met by trying to design classes that are in-line with good object-oriented guidelines. To this end, established design patterns are used in the system wherever possible (The reader is referred to *Gamma et al. (1995)* for a full treatment on design

patterns). For example, in order to meet the requirement for reuse, a generic framework for representing the problem domain is designed in order to reuse the functionality provided by the cost modeller infrastructure. This framework is based on the Factory and Template design patterns. The use of these patterns allows separation of domain knowledge from the infrastructure. This means that the problem domain being represented is able to change its structure completely without effecting the infrastructure.

In order to meet the requirement for flexibility, a modular approach is taken, in which system modules can be added or taken away, or individual modules can be replaced by a different implementation as long as the interface to the module remains consistent.

The current modular structure of the cost expert prototype is as shown in Figure 19. As can be seen from Figure 19, the cost expert application contains eight modules/components namely The CAD Link Module, The GUI Module, The Blackboard Module, The Case Base Module, The Case Based Reasoner Module, The Cost Estimator Module, The Design Interpreter Module and The Reporter Module. Let us begin by looking at the design of the Case Base Component and how problem domain knowledge is represented within it.

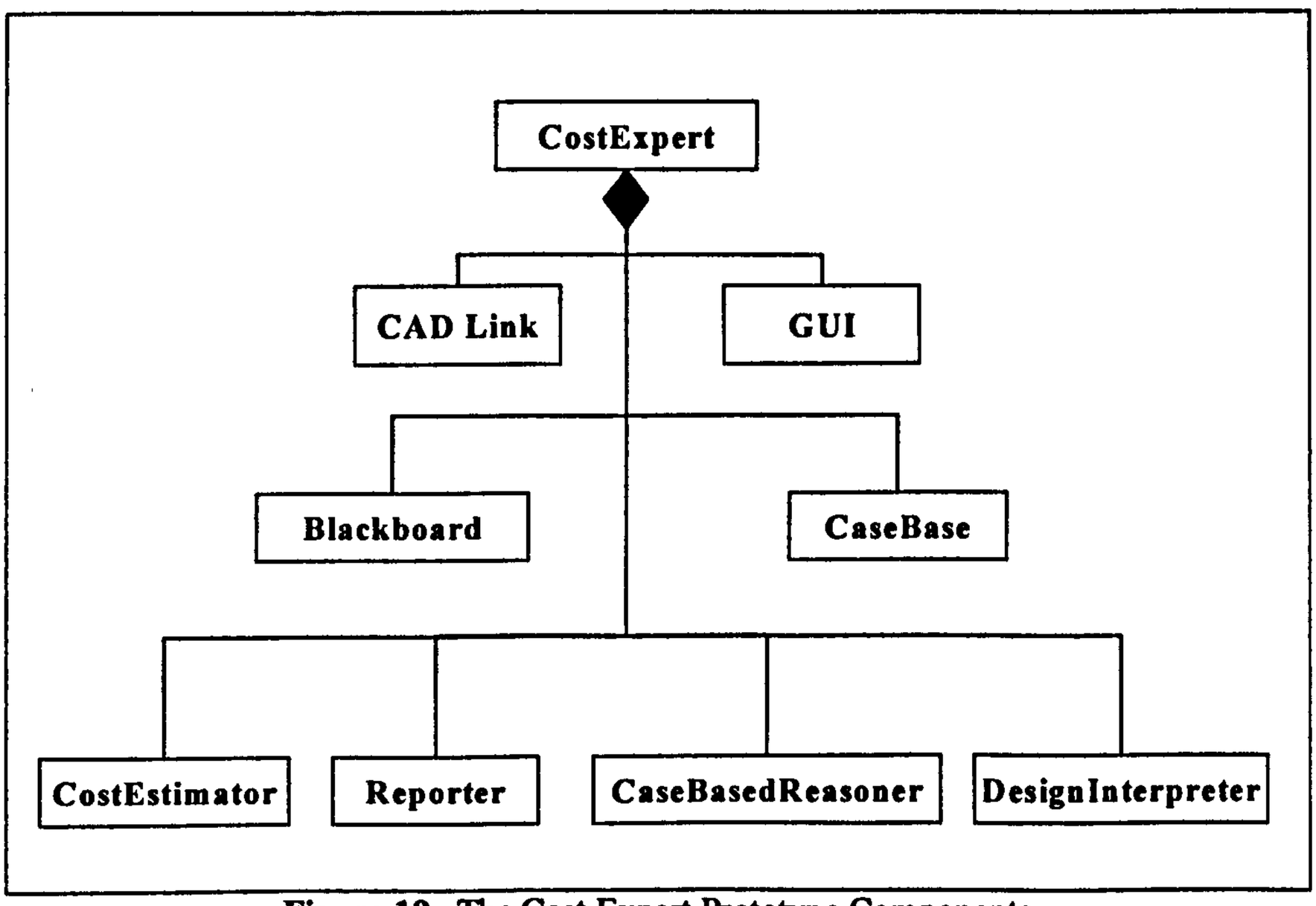


Figure 19. The Cost Expert Prototype Components

6.3 Knowledge Representation Strategy

The requirement for accessing persistent knowledge and data is fulfilled by the Case Base Module. The Case Base is responsible for representing and storing all persistent knowledge pertinent to the cost modelling process. Here, a generic framework for representing the problem domain is designed in which the structure of the problem domain is completely configurable. This is achieved by employing the Factory Class Creational Design Pattern.

The knowledge configuration process is as follows. Firstly, the problem domain under consideration has to undergo an object-oriented analysis. The objective of this analysis is to yield a frame-based representation of the problem domain. Here this representation is referred to as the Case Class Hierarchy. This is because all associations between problem domain entities are encapsulated with the exception of inheritance which is used to define the class that is instantiated. Figure 22 shows the Case Class Hierarchy used for the current case study. At the top of the hierarchy is the abstract Case Class. All problem domain entities inherit from this class and are therefore referred to as Cases. Once the Case Class Hierarchy is defined, a formatted Case Class File depicting this hierarchy and the corresponding Case Instances File containing the actual domain knowledge have to be prepared and that completes the knowledge configuration process.

Once the knowledge is configured, we are able to use the Case Base Module without modifying its structure or implementation. The actual structure of the Case Base is shown in Figure 20.

When the cost expert application is invoked, the first thing it does is initialise its components. The Case Base Component is initialised as follows. Referring to Figure 20, firstly, the *CaseFactory* is instantiated and associated with the *CaseBase*. On instantiation, the *CaseFactory* uses the *FileReader* to read in the formatted *CaseClassDefinitionsFile*, and the *CaseFactory* uses this definition to instantiate its set of Case Classes. Then the *CaseBase* uses the reader to read in the *CaseInstancesFile* and uses the *CaseFactory* to instantiate the *CaseBase*'s set of Case Instances.

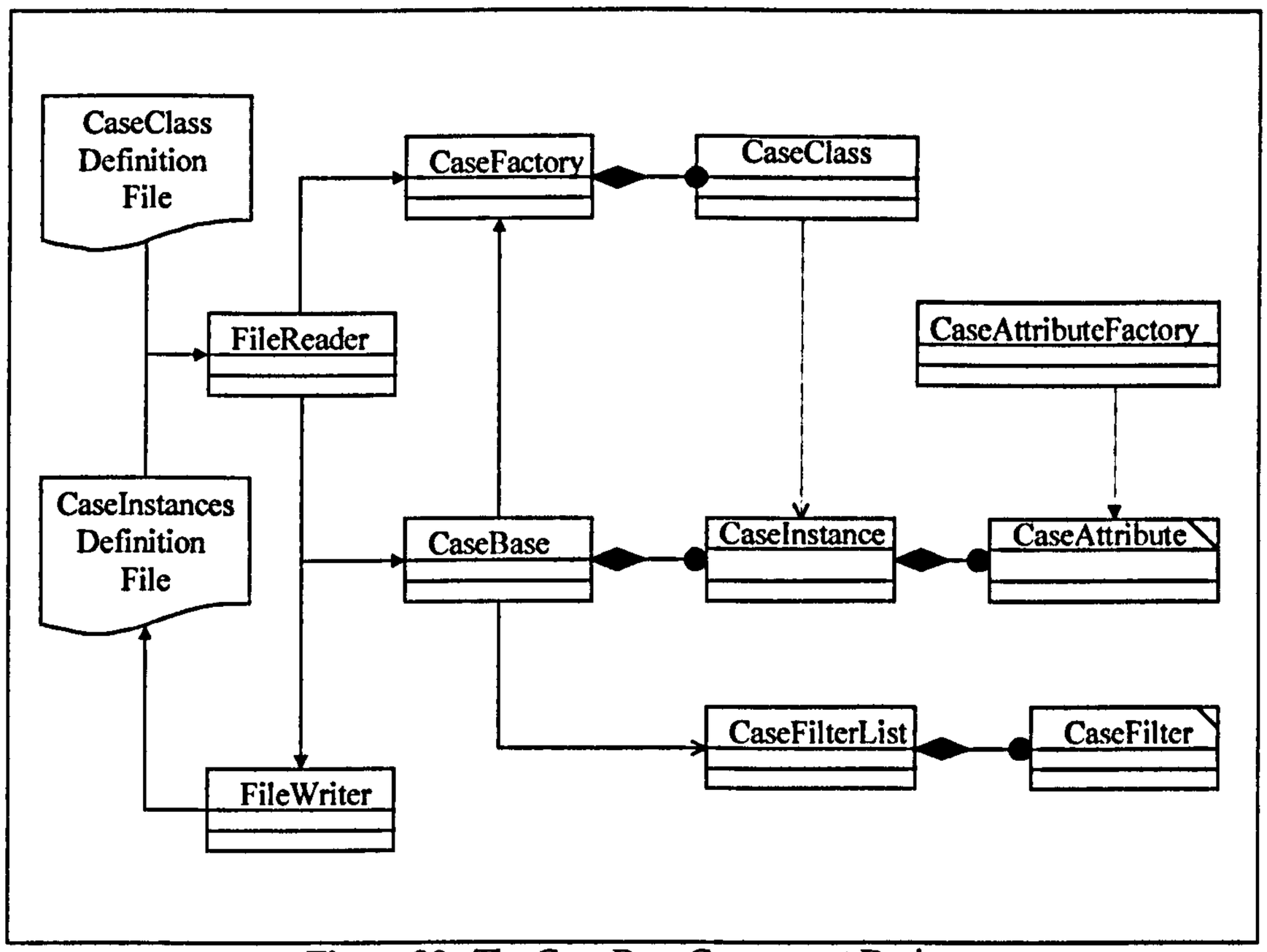


Figure 20. The Case Base Component Design

The Case Base itself is composed of one to many Case Instances which themselves are composed of one to many Case Attributes. The *CaseAttribute* class is an abstract class. The *CaseAttributeFactory* is employed to create the concrete derived case attributes. The derived case attribute classes defined currently are shown in figure 21.

The definition of the *CaseAttribute* structure employs the Template Behavioural Design Pattern. This design pattern supports specialisation and generalisation of behaviour between abstract and derived classes. The composition of a *Case* within the *CaseBase* is defined by its set of case attributes. Other than the basic attribute types such as the *Integer*, *Double* and *Long* attributes, we also have specialised attribute constructs, which are namely the *List*, *DataTable* and *Association* attributes. These are defined to increase the semantic richness of the representation schema. As mentioned before, all associations between problem domain entities are encapsulated bar inheritance. Here, all associations are represented by *CaseAssociationAttributes.* The *CaseAssociationAttribute* encapsulates all the parameters that may describe a particular association between cases such as the name of the association, the cardinality of the association and the members of the association.

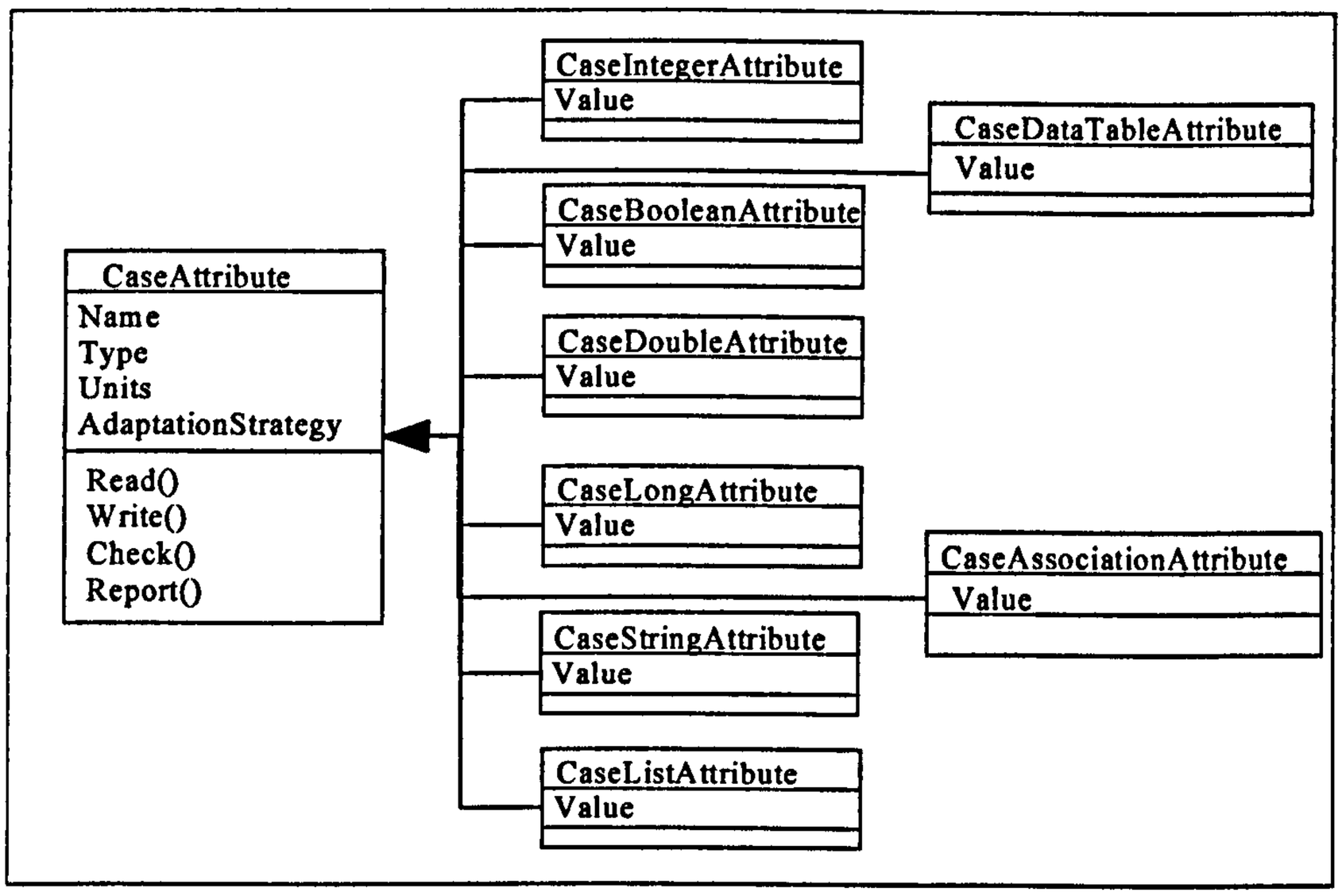


Figure 21. Case Attribute Definition

Figure 22 illustrates the Case Class Hierarchy defined for the current case study. What follows is an examination of some of the defined cases and their composition. Let us begin by looking at the composition of the Machine, Assembly and Part cases respectively. The designed methodology specifies the Case Base Structure as a 3-dimensional one that allowed multiple abstraction level browsing. This 3-D structure is depicted in the Case Class Hierarchy as follows. The 'has-a' construct relating different abstraction levels of the product design is modelled by the *ComponentList*, *AssemblyUses* and *MachineUses* association attributes. The 'is-a' construct relating a design case to a design cost model is modelled by the PartClass, AssemblyClass and MachineClass attributes, which are string attributes describing the class of components a particular design case belongs to. The actual case instances are created by the Case Factory from this Case Class Hierarchy definition and the Case Instances Definitions File.

Case browsing is implemented through the use of case filters. In figure 20, a *CaseFilterList* is depicted as being associated to the *CaseBase*. It is through this *CaseFilterList* that we are able to query the *CaseBase* to retrieve the cases of interest. A *CaseFilterList* is composed of one to many *CaseFilters*. The *CaseFilter* class itself is

an abstract class. The derived *CaseFilter* classes defined currently are the *CaseClassFilter* and the *CaseAttributeFilter*. The CaseClassFilter is a simple filter that retrieves only those cases of a specified class from the CaseBase. For example, one can specify a *CaseClassFilter* that retrieves all *Machine* cases from the *CaseBase*. The *CaseAttribute* filter is a more powerful filter that filters the *CaseBase* for cases that pass a particular condition placed on one of its *CaseAttributes*. The filtering conditions currently implemented are *GreaterThan*, *LessThan*, *Equals*, *NotEquals*, *IsWithinRange*, *IsMemberOf* and *Navigate*. The *IsMemberOf* and *Navigate* are special filtering conditions that apply to *CaseAssociationAttributes* only. The *IsMemberOf* condition checks to see if a *CaseAssociationAttribute* contains a specified *CaseInstance*. The *Navigate* condition specifies a navigation path through the CaseBase. It is this Navigation facility that allows multiple abstraction level browsing. For example, by specifying a navigation filter one is able to navigate from a *PartCase* to the set of *MachineCases* that the *PartCase* is a component of and filter this set further as necessary. The *CaseFilterList* itself acts as First In First Out (FIFO) Queue. That is that it applies the filters it contains to the *CaseBase* in strict linear order, removing each filter from the list as it is applied.

Returning to the composition of the *PartCase*, other than the *PartClass*, *AssemblyUses* and *MachineUses* attributes, the *PartCase* Class Definition consists of the *MaterialCost*, *SubContractCost*, *ManufacturingCost*, *AssemblyCost* and *FeatureModel* attributes, which are all defined as *CaseAssociationAttributes* with a cardinality of one. That is that each *PartCase* instance is associated with one instance of each of *MaterialCostCase*, *SubContractCostCase*, *ManufacturingCostCase*, *AssemblyCostCase* and *DesignComponentCase*. These cases along with the static data cases encapsulate the data necessary to represent the various cost contributors. Collectively, the case instances of the *MachiningCentreCase class*, *MaterialStaticData class*, *ManufacturingStaticData class*, *AssemblyStaticData class* and the *SubContractStaticData* class, model the production capabilities of the company. Finally, the *DesignComponentCase* class represents the physical realisation of the design. It contains the *ManufacturingFeatures CaseAssociationAttribute* whose members are the ManufacturingFeatureCase class instances that describe the design.

ALL MISSING PAGES ARE BLANK

IN

ORIGINAL

6.4 The Inference Engine

The inference engine is responsible for reasoning about the problem domain and moving the state of the system towards a solution state.

Reasoning within the inference engine is done by the knowledge modules. These are currently defined as:

- The *CaseBasedReasoner* Module, which contains the knowledge sources that can access and operate on the *CaseBase*.
- The *DesignInterpreter* Module, which contains the knowledge sources that can access the CAD model and generate a feature model for the selected design.
- The *CostEstimator* Module, which contains the knowledge sources that can produce a cost estimate for the current design; and,
- The *Reporter* Module, which contains the knowledge sources that report the explanation for the produced cost estimates to the user.

The activation of these knowledge sources is controlled by the control strategy defined by the Blackboard model of problem solving. Here this control strategy is encapsulated within the *Blackboard* Module. The blackboard module is responsible for storing the evolving solution and controlling the activation of the knowledge sources based on events that occur on the blackboard.

The top-level Cost Expert Dynamic Model illustrated in Figure 23 depicts how events are generated on the blackboard. Referring to Figure 23, the *CostExpert* application becomes active when invoked. The first thing it does in this state is initialise its system components. The last component to be initialised is the GUI component, which on initialisation enters the *CostExpert* into an X-Event loop. The X-Event loop is a continuous alert state imposed on the user interface which monitors any user input which is referred to as X-Events. The *CostExpert* therefore enters the state of waiting for X-Events. When the user activates a graphics widget (e.g. a GUI button), the *CostExpert* enters a state in which a GUI widget is active. In this state, it processes the associated callback function. The callback function defines what action to take should its associated widget be activated.

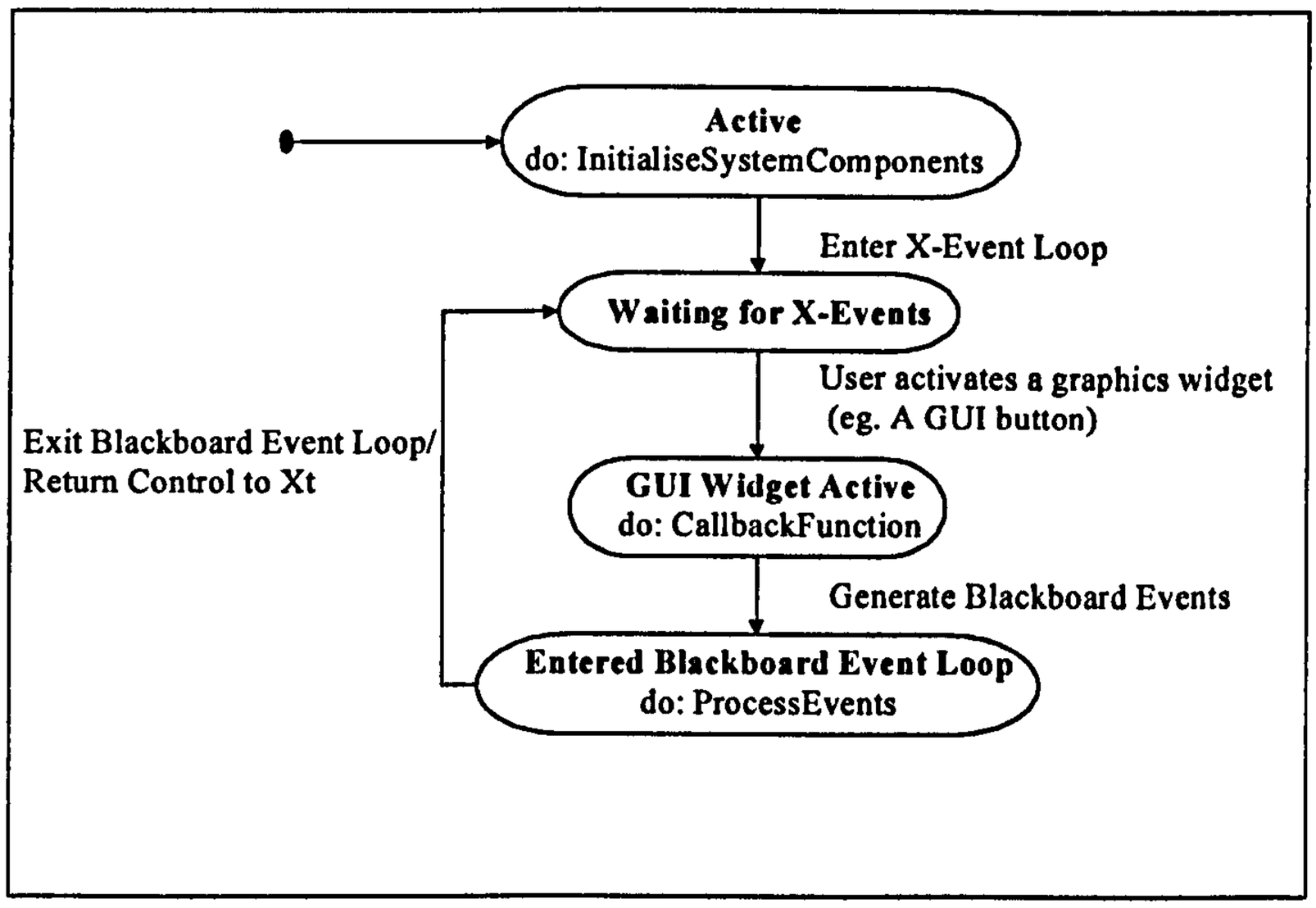


Figure 23. The Cost Expert Dynamic Model

In the *CostExpert* system, all GUI callbacks do nothing more than generate the appropriate events on the blackboard.

The *CostExpert* system then enters the Blackboard Event Loop. In this state, the blackboard control strategy takes control of the system. The blackboard recursively processes the generated events until no more blackboard events are generated at which stage the *CostExpert* exits the blackboard event loop and returns control to the X-Event loop. Through this approach, the *CostExpert* answers the requirement for being dynamic. That is that the *CostExpert* remains alert throughout the design phase and can therefore be activated by the designer at any point during the design. The CostExpert only initialises itself on initial activation, therefore the designer does not need to re-enter the design specification each time he requires a cost estimate.

Returning to the inference engine, Figure 24 illustrates the component usage structure within the inference engine. When the knowledge module system components are initialised, they register the events they can handle with the blackboard. The blackboard maintains an associative table of blackboard event types and the knowledge modules that have registered to be notified of these events.

When events are generated on the blackboard, the blackboard inserts the events in a list data structure. In the absence of priority being assigned to the generated events, events are inserted at the back of the *GeneratedEventsList* attribute of the *Blackboard.* The *Blackboard* treats this list as a First In First Out (FIFO) Queue. When the blackboard event loop is entered, its *ProcessEvents* protocol is invoked. In this protocol, the *GeneratedEventsList* is iterated through; the first event is taken off the queue and checked against the *EventHandlerTable*; the control of the problem solving process is then delegated to the associated knowledge module's *ProcessEvent* protocol. This protocol checks the event type and invokes the appropriate knowledge sources within it for processing that event. The *ProcessEvent* protocol may itself generate a list of blackboard events. If so, it notifies the *Blackboard* of these events before exiting and control is passed back to the *Blackboard.* And so, heuristic activation continues in the inference engine until the *GeneratedEventsList* is empty, at which stage control is passed back to the X-Event loop. In the component usage structure shown in Figure 24, the knowledge modules are depicted as sitting around the blackboard and have no internal attributes. This is in line with the blackboard model of problem solving, in which the knowledge sources are all able to access the evolving solution on the blackboard and update it as necessary.

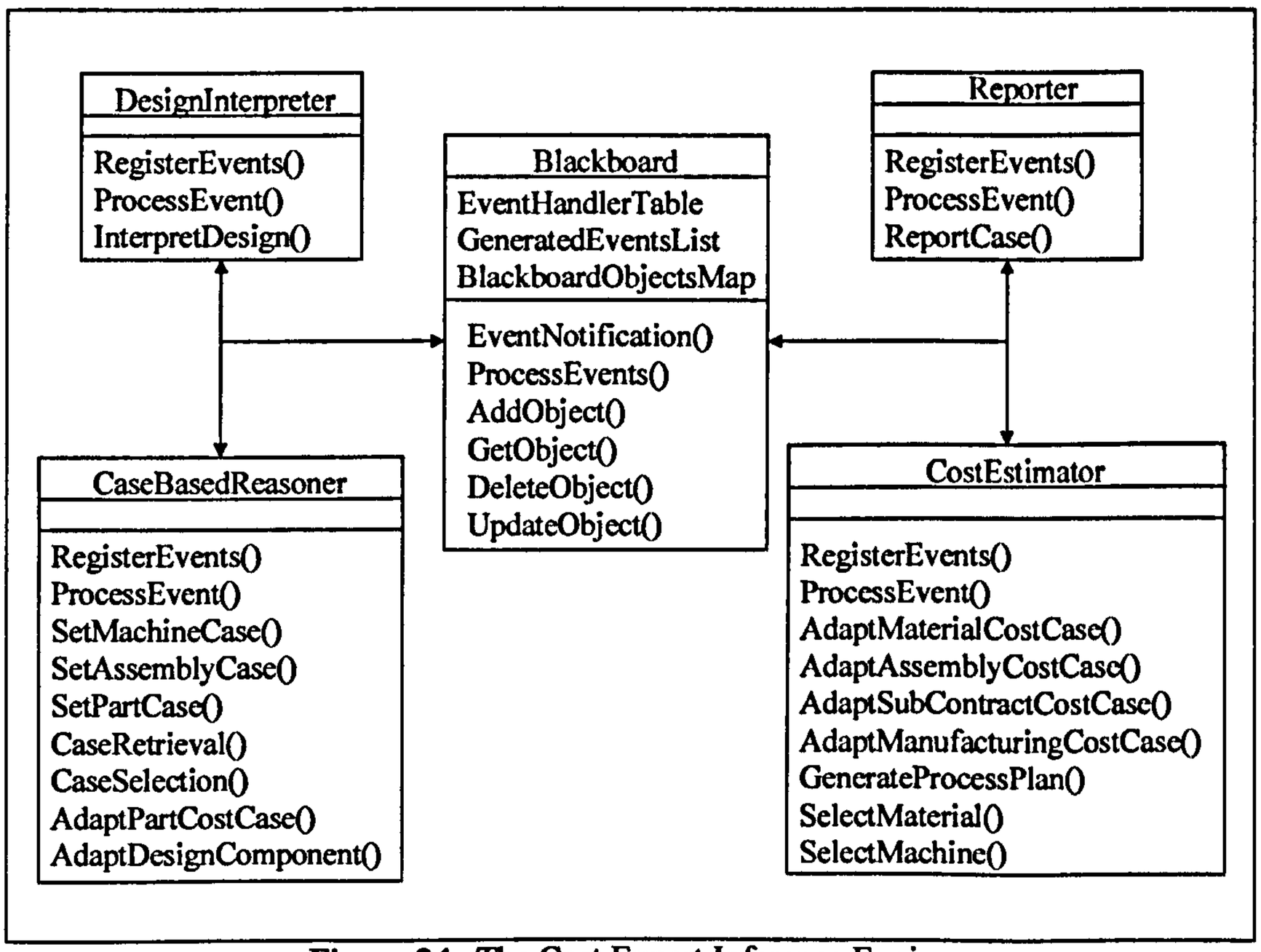


Figure 24. The Cost Expert Inference Engine

6.5 The Cost Expert Graphical User Interface (GUI)

The graphical user interface is developed in X-Windows and is X-Motif based. The GUI is designed so that it integrates well with the CAD system being used. The user then sees it as part of the CAD package and as it has the same look and feel as the CAD interface, the user is much more comfortable using it and won't see it as an interruption to his design session. Figures 25 and 26 illustrate the CAD interface and the Cost Expert interface respectively and show how the cost expert is invoked during a design session. The CAD interface in Figure 25 contains four main windows with which the designer interacts. These are the design window, in which the graphical design takes place; the icons window which contains the geometry creation tools that the designer uses; the prompt window that prompts the designer to take various actions; and, the list window to which any information that may be of interest to the designer is written. The prompt and the list windows are both employed by the Cost Expert as part of the user interface. This is because it is habit for the designer to look at these windows for information required or instructions to follow.

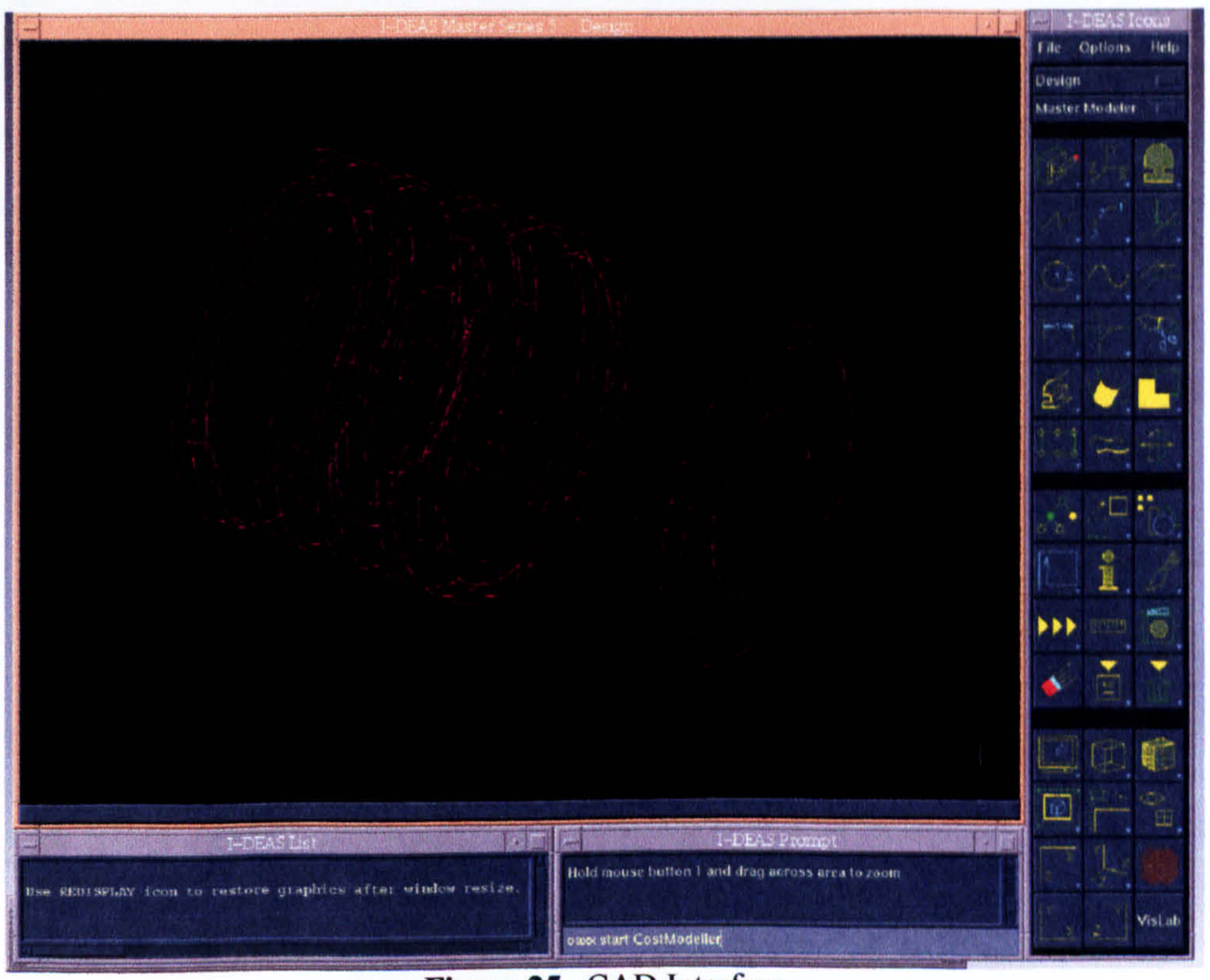


Figure 25. CAD Interface

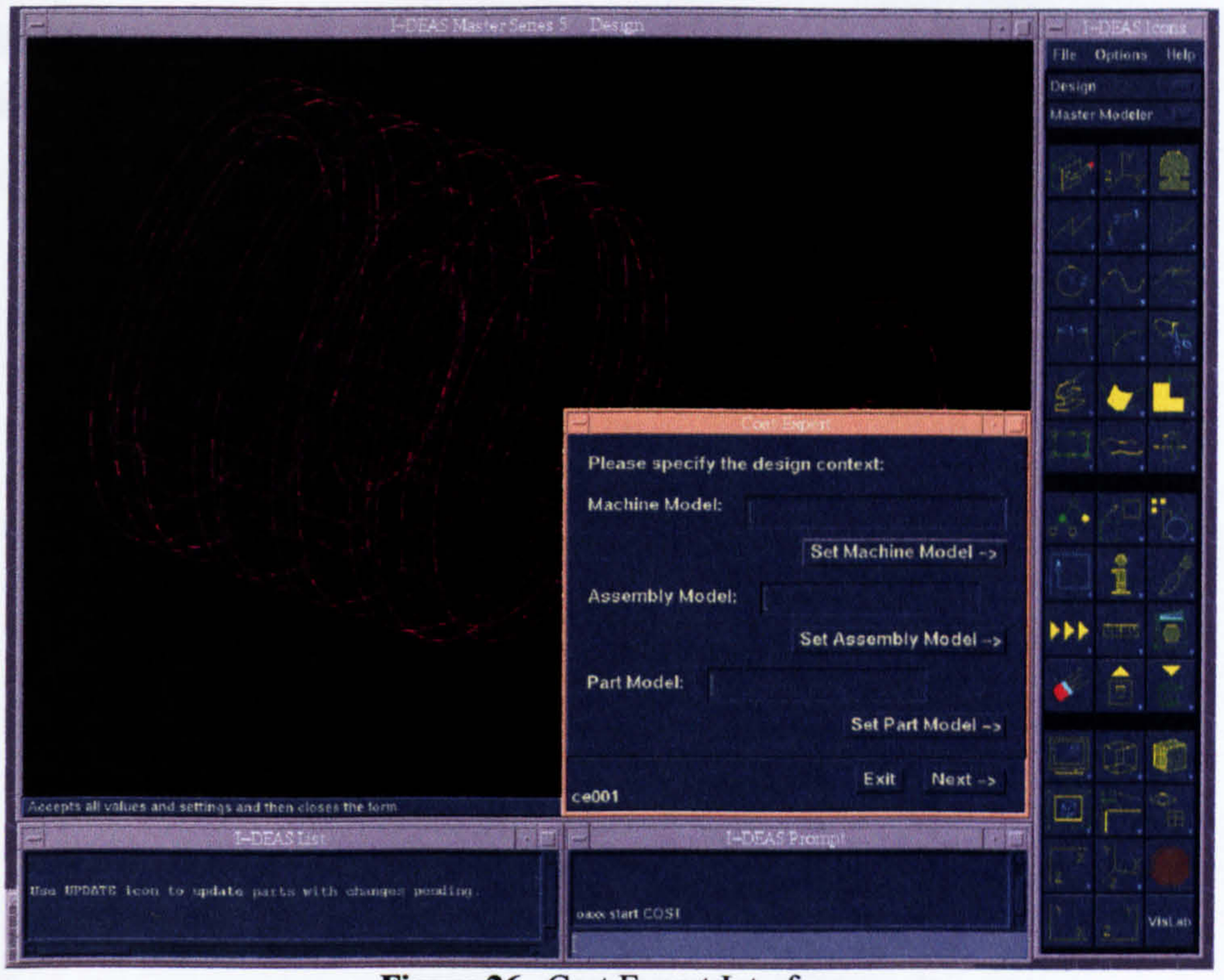


Figure 26. Cost Expert Interface

The initial screen that pops up when the cost modeller is invoked is shown in Figure 27. This screen shows a form for specifying the design context. The designer may enter any combination of the primary case keys, that is the machine model, assembly model and the part model. Once a model is set, its associated *Set Model* button is enabled. If the designer wishes to he may click on this button in order to specify secondary case keys.

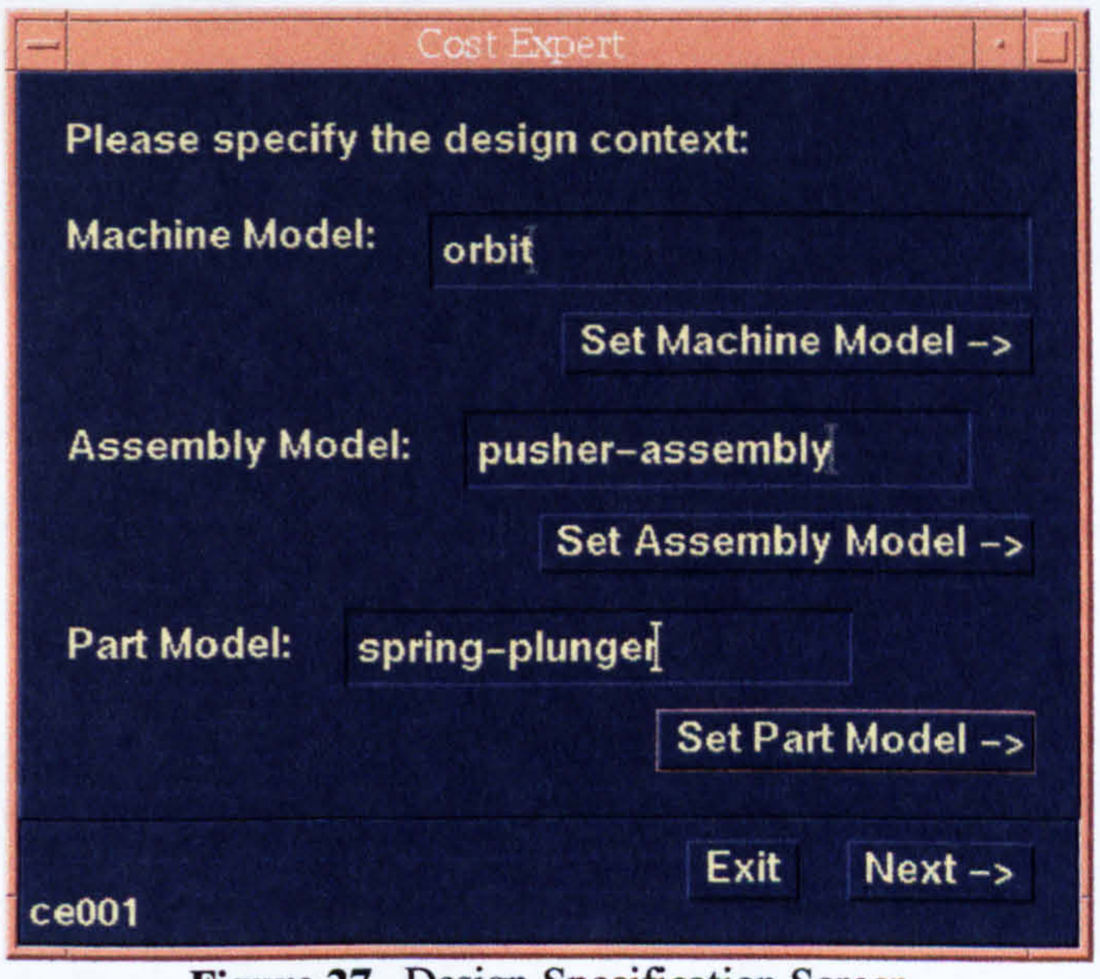


Figure 27. Design Specification Screen

Clicking on one of the set model buttons takes the designer to the index elaboration form shown in Figure 28. This screen shows the selected model's list of attributes. The designer may select any number of attributes he wishes to specify as secondary keys.

Figure 28. Index Elaboration Screen

Clicking on the *next* button takes the designer to screen ce003 where he may assign constraints and precedence order to the selected secondary indices. Clicking on the *back* button takes the designer back to screen ce002 where he can re-select the attributes he wants to define as secondary keys.

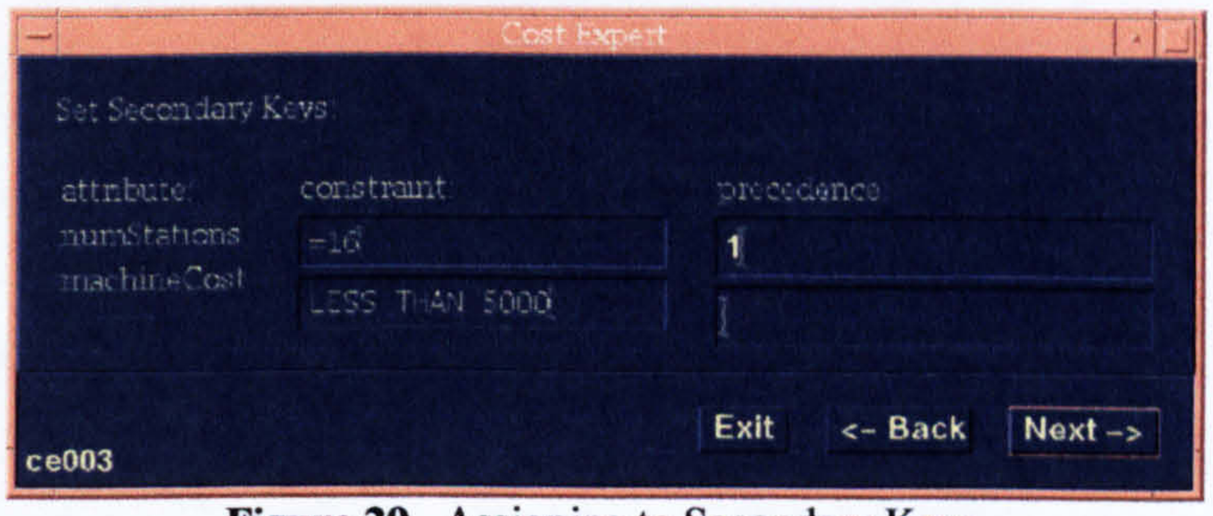
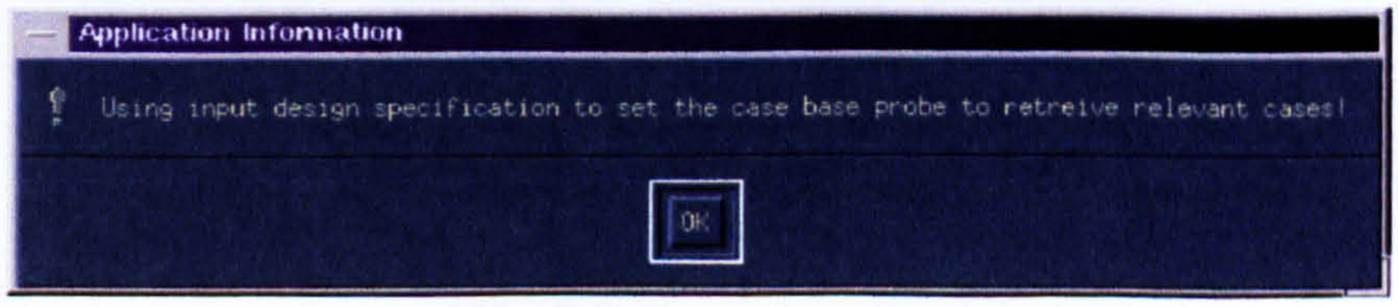


Figure 29. Assigning to Secondary Keys

Clicking on the next button on screen ce003 takes the user back to screen ce001 where he may specify the design further. Clicking on the next button on screen ce001 pops up the following information message:



This message is followed by screen ce004 shown in Figure 30, displaying the list of relevant cases and the selected closest matching case. The designer is free to browse the retrieved cases and change the selected case if he wishes to. He is also able to view any selected case by clicking on the *View Case* button, which brings up screen ce005 shown in Figure 31. This screen simply shows the design cost case selected. The designer is free to browse the case attributes.

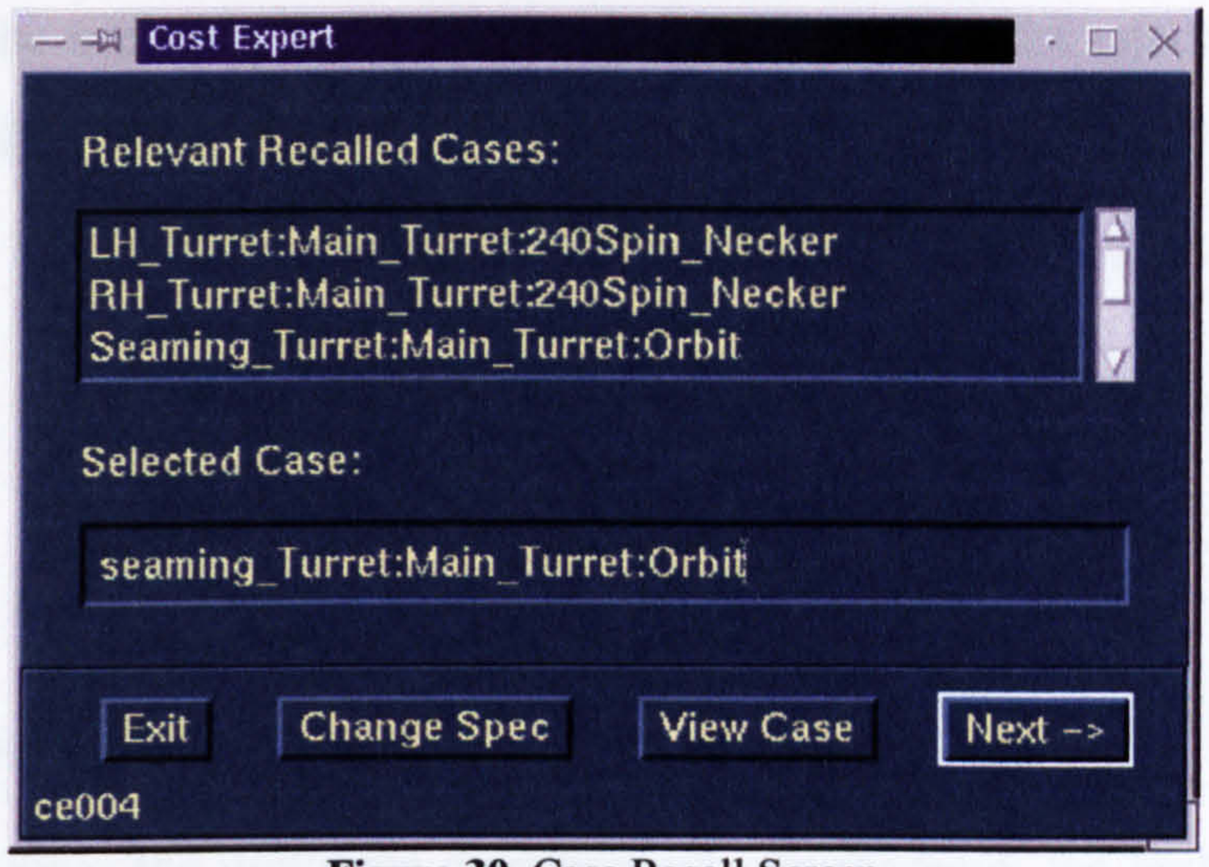


Figure 30. Case Recall Screen

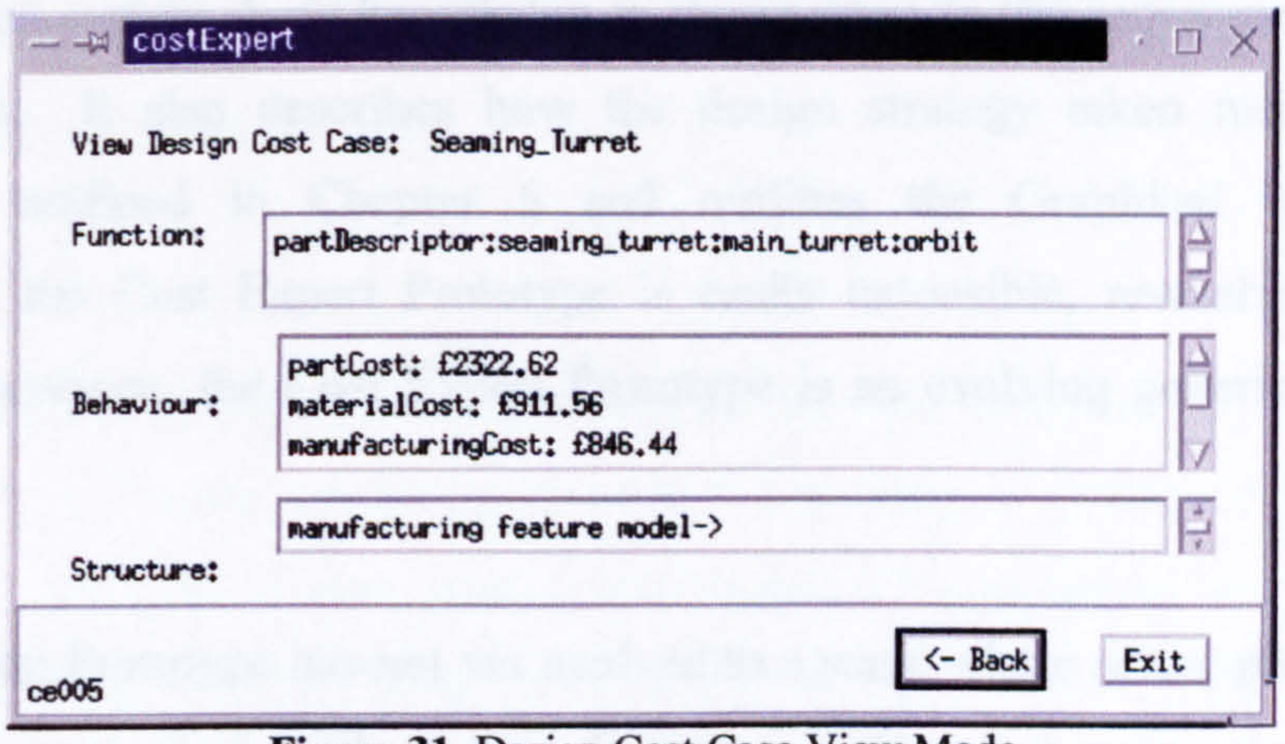
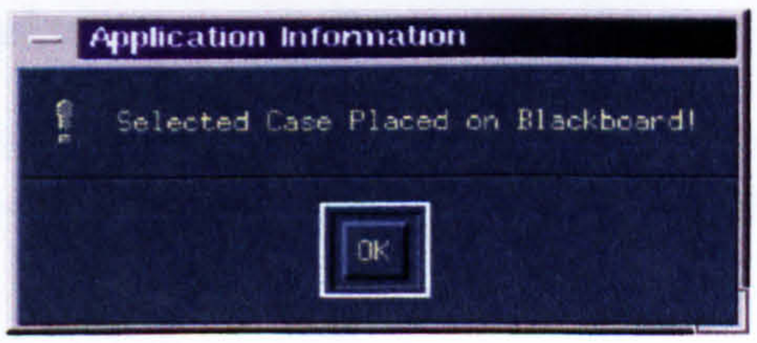


Figure 31. Design Cost Case View Mode

Clicking on the *Change Spec.* button on screen ce004 takes the designer back to screen ce001 where he is able to re-specify the design context. Clicking on *next* button pops up the following information message:



This message is followed by screen ce006 shown in Figure 32. This screen allows the designer to decide which action to take next. If the designer decides to pause for design, he may do so by clicking on the relevant key, which iconifies the *cost expert* window until it is needed. Clicking on the *change spec.* key returns the designer to screen ce001. Clicking on *exit* at any time during the *cost expert* session exits the cost modeller facility. Clicking on any one of the cost part, cost assembly, cost machine keys transfers the user interface to the I-DEAS prompt and list regions for query and explanation, respectively.

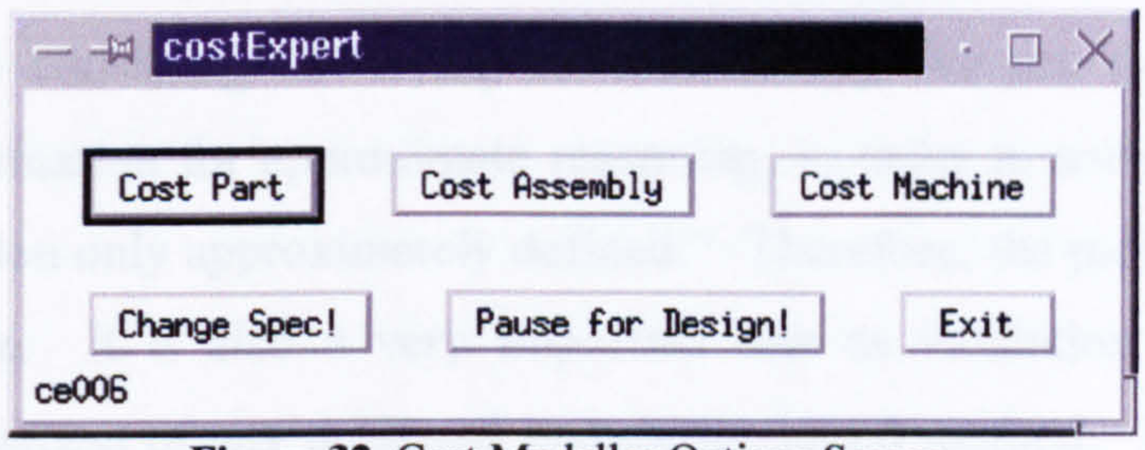


Figure 32. Cost Modeller Options Screen.

6.7 Chapter Summary

This chapter outlines the prototype development environment, the high-level design of the Cost Expert system, how knowledge is represented in the prototype and how it is reasoned with. It also describes how the design strategy taken meets the system requirements outlined in Chapter 5 and outlines the Graphical User Interface. Summarising, the Cost Expert Prototype is easily extensible, re-usable, flexible and dynamic. In essence, the Cost Expert Prototype is an evolving generic shell for cost modelling.

The Cost Expert Prototype has not yet evolved to a stage where actual product costs are generated for comparison with real product costs. This is because the focus of this research study has been the design and implementation of a generic framework for cost modelling and not on knowledge capture and refinement. The knowledge incorporated into the prototype from the current case study, however, is of sufficient quality to allow logic traces of the cost estimation process to be examined.

7 Validation, Verification and Test

7.1 The Validation Problem

The problem of validating a knowledge-based system is described well by *Ayel and Laurent(1991)* as consisting of "trying to validate approximate systems which use approximate information for approximate reasoning, in order to solve problems which are themselves often only approximately defined." Therefore, the problem of validation is a complex one. It is also a very important one as validation is central to the acceptance and success of any proposed knowledge-based method. The research field of validation, verification and test is still an immature and uncoordinated one, consisting of isolated research efforts aimed at individual knowledge-based systems. Widely accepted generalised methods of validation are yet to emerge. The reader is referred to *Ayel and Laurent(1991)* who provide an extensive literature survey of the validation, verification and test research field

7.2 VVT Plan

As what has been developed is a prototype system, the tests that are defined for the prototype must be limited by the extent of the knowledge population of the prototype. Here, we are testing a modular system, therefore each module of the system can be tested independently from the others, this allows for easier system validation. The test plan for validating the cost modeller prototype is as follows:

1. Testing the Case-Based Reasoning Facility
   a. Test case retrieval

      For three different test case specifications, check to see that all relevant cases are retrieved. In particular, set constraints and priorities to secondary case keys for testing for correct search space reduction.

b. Test case selection

For three different test case specifications, check to see that the closest matching case is selected correctly.

c. Test case adaptation

For three different test cases, check to see

- Firstly, that if there is no actual design done and a specification is entered, then the parametric cost estimation knowledge is used for adaptation. Check to see that the case adaptation is correct.
- Secondly, check to see that if actual design has been done then the case adaptation is done by the combination of substitution/transformation specified. Check to see that the case adaptation is correct.

2. Testing the Feature Recognition Algorithm

For three different CAD designs (one mainly milled component, one small rotational component with profile features along its diameter and one large rotational component) perform the following checks:

a. Check that the history tree associated with the part is traversed in the correct order as defined by the Feature Recognition Rule Set.
b. Check to see that the correct 2-D features are recognised from the wireframes associated with the leaf nodes of the history tree.
c. Check to see that the recognised 2-D features are mapped correctly to their associated 3-D features.

3. Testing the Cost Knowledge Module

For three different detail designs (one mainly milled component, one small rotational component with profile features along its diameter and one large rotational component) perform the following checks:

a. check that the correct material is selected;
b. check that the correct machining centre is selected; and,
c. check that cost rules are applied correctly by comparing produced unit costs with the expert's cost estimates.

4. Testing the communication protocol

   For each module and its parts test that, they are able to register for event notification and that they are correctly notified of events that occur on the blackboard.

5. Testing the entire infrastructure

For three design cost specifications, test that

a. the relevant cases are recalled;
b. the closest matching case is selected;
c. the case adaptation is performed correctly; and,
d. the derived costs are as expected.

4.3 Results

Test	Verified by:	Result : as expected(√)
1a. Case Retrieval	observation	√
1b. Case Selection	observation	√
1c. Case Adaptation	observation	√
2. Feature Recognition	observation	√
2a. Tree Traversal	observation	√
2b. 2-D FR	observation	√
2c. Feature Mapping	observation	√
3a. Material Selection	observation	√
3b. M/C Selection	observation	√
3c. Cost Estimation	observation	√
4. Communication Protocol	observation	√
5. Infrastructure	observation	√
5a. Case retrieval	observation	√
5b. Case Selection	observation	√
5c. Case Adaptation	observation	√
5d. Cost Derivation	observation	√

7.4 Chapter Summary

This chapter has outlined the tests carried out on the prototype. The results of the tests are evaluated through observation of the cost modeller prototype run. The tests done are not exhaustive as they are limited by the extent of the prototype's knowledge base.

However, the tests carried out show the prototype to behave as expected from the definition of the cost modelling methodology.

---

8 Discussion, Conclusions and Further Work

8.1 Discussion

The mission statement for this research study was to address the issues relating to the problem of knowledge-based cost modelling for innovative design and to design a methodology that resolves these issues. The methodology was also required to address the issue of cost modelling throughout the design phase, from conceptual to detail design. In response to these requirements, a novel method for cost modelling is devised in this research study.

The main issues were identified in this research as those of design completion, design interpretation and the capture and application of cost heuristics. Each one of these issues is realised to address the problem domain at different abstraction levels, each abstraction level being very loosely coupled to the other abstraction levels, and a modular infrastructure is imposed on the problem domain. This modular infrastructure is imposed for the reasons of managing problem-domain complexity, easing the validation, verification and test process and increasing the system modularity in order to provide a robust and flexible infrastructure. Each module in the infrastructure is then treated as a black box that can receive and process the interactions it is designed to handle.

In complex problem domains, such as cost modelling, the interactions within modules as well as those between modules can prove to be difficult to manage. For this reason, a communication protocol is introduced into the infrastructure that handles all communication between the modules. The communication protocol defined is one based on the blackboard method of problem solving. This is because the modular nature of the defined infrastructure makes it suitable for placement directly within a blackboard architecture for which established communication protocols are defined.

As each module in the infrastructure is treated as a black box, the internal specification of a particular module can be done independently from the other modules. That is that the most appropriate approach for resolving each of the identified issues can be devised and adopted in the cost modelling strategy.

A case-based reasoning approach is devised in the methodology to address the issue of design completion. The issue of design completion has to be resolved if the methodology is to model costs throughout the design phase. The case-based reasoning approach taken involves recalling design cost cases from a design cost case base. These cases as the name suggests may contain both design knowledge and cost knowledge depending upon specification. Designs cost models are defined for specifying the content of design cost cases that are analysed to be of the same type. Design cost case adaptation is done by a combination of substitution, derivation and transformation.

A feature-recognition approach is devised in the methodology to address the issue of design interpretation. The issue of design interpretation also has to be resolved if the methodology is to model costs throughout the design phase. The feature recognition approach taken is one of hybrid rule-based automatic feature recognition. The devised method operates on a hybrid geometry representation, that is a combination of CSG, sweep and wireframe models. Feature recognition proceeds by traversing the history tree and querying the tree nodes in order to extract the features associated with each node. The feature extraction is based on a search for pre-defined 2-D features in the design node's wireframe representation followed by a heuristic mapping that maps the recognised 2-D features to the associated 3-D manufacturing features.

Finally, an object-oriented approach is taken in the methodology for representing and structuring cost knowledge.

8.2 Conclusions

Concluding, the contribution to new knowledge from this research study is a novel method for modelling product costs throughout the design phase for innovative design activity. The methodology is validated through a prototype. From the test results

carried out on the developed prototype, the prototype is seen to produce the expected results. Although the test set is a small one limited to the prototype's populated knowledge, it still provides confirmation that the designed methodology is a feasible one.

8.3 Suggestions for Further Work

The following are a few suggestions for further work and prototype development. Firstly, automating case collection must be investigated. In domains such as machine design in which there are a large number of case instances, manual case collection can be a mundane and time-consuming task. In particular, the manual preparation of the formatted case instances file required by the developed cost expert prototype is highly prone to human error. Therefore, automating case collection is a necessary future development.

Secondly, different case studies need to be examined in order to study how generic the developed prototype is, and where possible areas identified in which the prototype can become more configurable.

The focus of this research study has been the design and implementation of a generic framework for cost modelling. The objective of the knowledge capture here has been to capture knowledge from the current case study of sufficient quality to allow logic traces of the cost estimation to be examined. However, the captured knowledge is not yet refined enough for real cost comparisons to be done. The focus in the future needs to shift to effective knowledge capture and knowledge refinement so that real cost comparisons can be carried out and the captured domain knowledge evaluated.

Another possible future development is the modelling of uncertainty that may be associated with a produced cost estimate. Also further consideration should be given to how the blackboard system can incorporate a horizontal dimension for storing and reasoning with intermediate design solution states. Finally, we have consistency checking, which can realistically be done only on a complete system.

ALL MISSING PAGES ARE BLANK

IN

ORIGINAL

REFERENCES

1 Andreason M. M, Kahler S. and Lund T. (1983a). *Design for Assembly*, IFS Publications Ltd., Bedford.

2 Apgar H. E. and Daschbach J. M. (1987). *Analysis of Design through Parametric Cost Estimation Techniques.* In *Proceedings of the International Conference on Engineering Design*, Boston, MA, pp. 759-766.

3 Ayel M. and Laurent J-P. (1991). *SACCO-SYCOJET: Two Different Ways of Verifying Knowledge-Based Systems. In Validation, Verification and Test of Knowledge-Based Systems* (Eds. Ayel M. and Laurent J. P.), John Wiley and Sons, pp. 63-75.

4 Boothroyd G. and Reynolds, C. (1989). Journal of Manufacturing Systems 8(3), pp.191.

5 Boothroyd G., Dewhurst P. and Knight W. (1994). *Product Design for Manufacture and Assembly*, Marcel Dekker, Inc.

6 Buchanan B. G. and Shortliffe E. H. (1984). *Rule-Based Expert Systems*, Addison-Wesley, Reading, MA.

7 Carr D. K. and Johansson H. J. (1995). *Best Practices in Reengineering*, McGraw-Hill, Inc.

8 Choi B. K., Barash M. and Andersen D.C. (1984). *Automatic Recognition of Machined Surfaces from a 3D Solid Model.* Computer-Aided Design 16(2) pp81-86.

9 Coad P. and Yourdon E. (1991). *Object-Oriented Analysis*, 2nd edition. Englewood Cliffs, NJ: Yourdon Press/Prentice-Hall.

10 Coad P. and Yourdon E. (1991). *Object-Oriented Design.* Englewood Cliffs, NJ: Yourdon Press/Prentice-Hall.

11 Corbett J. (1986) Annals of CIRP, 35(1), pp. 93.

12 Craig I. D. (1991) *Formal Specification of Advanced AI Architectures.* LS Horwood, NY.

13 Chuang S. H. (1991). *Feature Recognition from Solid Models Using Conceptual Shape Graphs.* Ph.D. Arizona State University.

14 Dewhurst P. and Boothroyd G. (1987). *Early Cost Estimating in Product Design.*

15 Domeshek E. A. and Kolodner J. L. (1997). *The Designers' Muse: Experience to aid conceptual design of complex artefacts.* In *Issues and Applications of Case-Based Reasoning in Design* (Eds. Maher M. L. and Pu P.), Lawrence Erlbaum Associates, NJ, pp. 11-36.

16 Dym C. and Levitt C. (1991). *Knowledge-Based Systems in Engineering*, McGraw-Hill, New York.

17 Evbuomwan N. F. O., Sivaloganathan S. and Jebb A. (1996). *A Survey of Design Philosophies, Models, Methods and Systems.* In *Journal of Engineering Manufacture Part B, pp.*301–320.

18 Gamma E., Helm R., Johnson R., Vlissides J. (1995). *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley.

19 Gonzalez A. J. and Dankel D. D. (1993). *The Engineering of Knowledge-Based Systems (Theory and Practise).* Prentice Hall.

20 Goel A. K., Bhatta S. R and Stroulia E. (1997). *KRITIK: An Early Case-Based Design System.* In *Issues and Applications of Case-Based Reasoning in Design* (Eds. Maher M. L. and Pu P.), Lawrence Erlbaum Associates, NJ, pp. 87-129.

21 Goel A and Kolodner J. L (1991*). Towards a case-based tool for aiding conceptual design problem solving. Proceedings of the DARPA Workshop on Case Based reasoning.* Morgan Kaufmann pp 109-120.

22 Graham I. M. (1994). *Object-Oriented Methods.* Second Edition. Wokingham:Addison-Wesley.

23 Hammond K (1986). *CHEF: A model of case-based planning. Proceedings of the AAAI '86.*

24 Hart A. (1986). *Knowledge Acquisition for Expert Systems.* ISBN: 185091091X.

25 Henderson M. R., Gopal S., Stage R, Walker K and Regli W. (1994*). Boundary Representation-Based Feature Identification.* Advances in Feature Based Manufacturing by J.J. Shah, M. Mantyla and DS Nau (Editors). Elsevier Science B.V.

26 Hinrichs T R and Kolodner JL (1991). *The role of adaption in case based design, Proceedings of the DARPA Workshop on Case-Based Reasoning.* Morgan Kaufmann pp 121-132

27 Hoult D. P. and Lawrence Meader C. (1996). *Methods of Integrating Design and Cost Information to Achieve Enhanced Manufacturing Cost/Performance Tradeoffs.* In *SAVE International Conference Proceedings.*

28 Hoult D. P. and Lawrence Meader C. (1996). *Predicting Product Manufacturing Costs from Design Attributes: A Complexity Theory Approach*, Society of Automative Engineers Publications Group, USA. Publication no. 960003.

29 Humphreys K. K. (1995). *Project and Cost Engineer's Handbook* (third Edition), New York: Marcel Dekker.

30 Humphreys K. K. (1996). *Basic Cost Engineering* (third Edition), New York: Marcel Dekker.

31 Joshi S. and Chang T. C. (1988). *Graph-Based Heuristics for Recognition of Machined Features from a 3D Solid Model.* In Computer-Aided Design, Vol. 20, no. 2.

32 Kolodner J.L. (1993). *Case-based Reasoning*. Morgan Kaufmann.

33 Koton P.(1988). *Integrating case-based and causal reasonings, Proceedings of the Tenth Annual Conference of the Cognitive Science Society.* Hillsdale,NJ. Lawrence Earlbaum Associates.

34 Knight W. A. and Poli C. (1985). *A Systematic Approach to Forging Design.* In *Machine Design* 57 (January 24), pp. 94-99.

35 Kyprianou L. (1980). *Shape Classification in Computer Aided Design.* Ph.D., Cambridge University, UK.

36 London P., Hankins B., Sapposnek M. and Luby S. (1987). In *Proceedings of the 1987 ASME International Computers in Engineering Conference and Exhibition* (Eds. Raghavan R. and Cokonis T. J.), ASME, New York, pp. 125-129.

37 Maher L. M. and Balachandran M. B. (1994b). *A multimedia approach to case-based structural design.* ASCE Journal of Computing in Civil Engineering 8(3) pp 359-377 .

38 Maher L. M., Balachandran M. B. and Zhang D. M. (1995). *Case-Based Reasoning in Design*, Lawrence Erlbaum Associates, NJ.

39 Maher M. L and Zhang D. M. (1993). *CADSYN: A case-based design process model.* Artificial Intelligence for Engineering Design, Analysis and Manufacturing 7(2):pp97-110.

40 Narasimhan S., Sycara K. P. and Navin-Chandra D. (1997). *Representation and Synthesis of Mechanical Devices.* In *Issues and Applications of Case-Based Reasoning in Design* (Eds. Maher M. L. and Pu P.), Lawrence Erlbaum Associates, NJ, pp. 187-218.

41 Navinchandra D. (1988). *Case-based reasoning in CYCLOPS, a design problem solver.* Proceedings of the DARPA Workshop on Case-based reasoning. Morgan Kaufmann pp 286-301.

42 Nieminen J. and Tuomi J. (1991). *Design with features for manufacturing cost analysis* in *Product Modeling for Computer-Aided Design and Manufacturing* (Eds. Turner J., Pegna J. and Wozny M.), Elsevier Science Publishers B.V.

43 Opas J. and Mantyla M. (1994). *Feature-based part programming.* In *Advances in Feature-Based Manufacturing* (Eds. Shah J. J., Mantyla M. and Nau D. S.), Elsevier Science B.V.

44 Peters T. J. (1991*). Encoding Mechanical Design Features for Recognition Via Neural Nets (Technical Report No. CSE-TR-91-20).* Department of Computer Science, University of Connecticut.

45 Poli C., Fengolio F. and Shunmugasundaram S. (1991). Concurrent Engineering 1 (March/April), pp.31-38.

46 Regli W. C. and Nau D.S. (1993). *Building a General Approach to Feature Recognition of Mechanical Parts.*

47 Rumbaugh J. et al. (1991). Object-Oriented Modelling and Design. Englewood Cliffs, NJ: Prentice-Hall.

48 Sakurai H. and Chin C-W. (1994). *Definition and Recognition of Volume Features for Process Planning.* In *Advances in Feature-Based Manufacturing* (Eds. Shah J. J., Mantyla M. and Nau D. S.), Elsevier Science B.V.

49 Shah J. J. and Mantyla M. (1995). *Parametric and Feature-Based CAD/CAM*, John Wiley and Sons, Inc.

50 Sycara K. P., Navinchandra D and Narasimhan S (1992*). Parametric adaption case based design.* Proceedings of the Case-Based Design Workshop, AID '92, pp113-125.

51 Taylor I. M. (1997). *Cost Engineering - A Feature Based Approach.* In the 85th Meeting of the AGARD Structures and Material Panel, Aalborg, Denmark, 14:1-9.

52 Tsatsoulis C. and Alexander P. (1997). *Integrating Cases, Subcases, and Generic Prototypes for Design.* In *Issues and Applications of Case-Based Reasoning in Design* (Eds. Maher M. L. and Pu P.), Lawrence Erlbaum Associates, NJ, pp. 261-297.

53 Vandenbrande J. H. (1990). *Automatic Recognition of Machinable Features in Solid Models. (Ph.D Thesis published as Technical Report No IRIS #260).* Computer Science Department and Institute for Robotics and Intelligent Systems, University of Southern California.

54 Vandenbrande J. H. and Requicha A. A. G. (1993). *Spatial Reasoning for the Automatic Recognition of Machinable Features in Solid Models.* In IEEE Transaction on Pattern Analysis and Machine Intelligence, Vol. 15, no. 12, pp. 1269-1285.

55 Vob A and et al (1994). *Retrieval of similar layouts, in J S Gerso (ed).* Artificial Intelligence in Design '94. Dordrecht: Kluwer Academic, pp 625-640.

56 Venkatachalam A. R., Mellichamp J. M. and Miller D. M. (1991). *Automating Design for Manufacturing through Expert Systems Approaches.* In *Design for Manufacture: Strategies, Principles and Techniques* (Eds. Corbett J. et al.), Wokingham: Addison-Wesley.

57 Venkatachalam, A. R. (1990). *A Knowledge-Based Approach to Design for Manufacturability, Ph.D. Dissertation,* The University of Alabama.

58 Vogt Jr., C. F. (1988). *Beyond CAD and CAM: Design for Manufacturability.* In *Design News* 44 (March 7): 18.

59 Whitney, D. E. (1988). Harvard Business Review 66 (July-August): 83.

60 Wozny M. J., Pratt M. J., Poli C. (1994). *Topics In Feature-Based Design and Manufacturing.* In *Advances in Feature-Based Manufacturing* (Eds. Shah J. J., Mantyla M. and Nau D. S.), Elsevier Science B.V.

Publications from this Research

1. Rehman S. and Morris A. J. (1996). A Knowledge-Based Approach to Modelling Manufacturing Costs at the Design Phase of a Product's life-cycle. In *Advances in Design and Manufacturing*, Vol. 7, pp. 215-225. ISSN number: 0926-9622.
2. Rehman S. and Guenov M. D. (1998). *An Artificial Intelligence Approach To Innovative Design Cost Estimation.* In the *Aerogram,* Vol. 9 Number 2, ISSN number: 0265-8569.
3. Rehman S. (1998). *A Methodology for Modelling Manufacturing Costs at Conceptual Design.* In Computers And Industrial Engineering, Vol. 35:3-4, pp. 623-626. ISSN number: 0360-8352.

APPENDIX A -Case Study

A1 Case Study Outline

The industrial case study in machine design considered in this research is carried out at CarnaudMetalbox (CMB) a company that specialises in the design and manufacture of can-making machines. CMB is a medium sized enterprise that is in the process of reengineering its practices. This research project was initiated in the early stages of the reengineering effort where the management identified cost estimation at the design phase as being one of the processes requiring reengineering.

The design activity at CMB mainly involves the designing of prototype machines. The designed machines may either be a variation on an existing family of machines or a completely new machine. The various phases that a design goes through at CMB are outlined in Figure a-1.

Firstly, market research and customer specification establishes the requirements for a new machine. The initial specifications from this phase include brief specification of the machine function, target costs and time scales. Once the requirement for a new machine has been identified, a feasibility study is carried out. At this phase, the focus is much more strongly on the function of the new machine. The initial analysis carried out is a very crude one and simply leads to the knowledge of whether or not it is possible to meet the specified function requirements, within the constraints set out in the initial specification. On establishing the feasibility of producing the new machine, the designer considers whether any unnecessary functions and costs can be eliminated from the original crude analysis. At this stage, the planning department is consulted for advice on manufacturability and the possible costs of production. The revised design specification is then outlined.

At the detail design phase, much of the design process is governed by 'rules of thumb' based on the individual designer's experience. The designers at CMB practice value engineering that is that they follow guidelines of design for economic manufacture that

have been empirically derived from years of design and manufacturing experience. Following these guidelines generally leads to improved manufacturability.

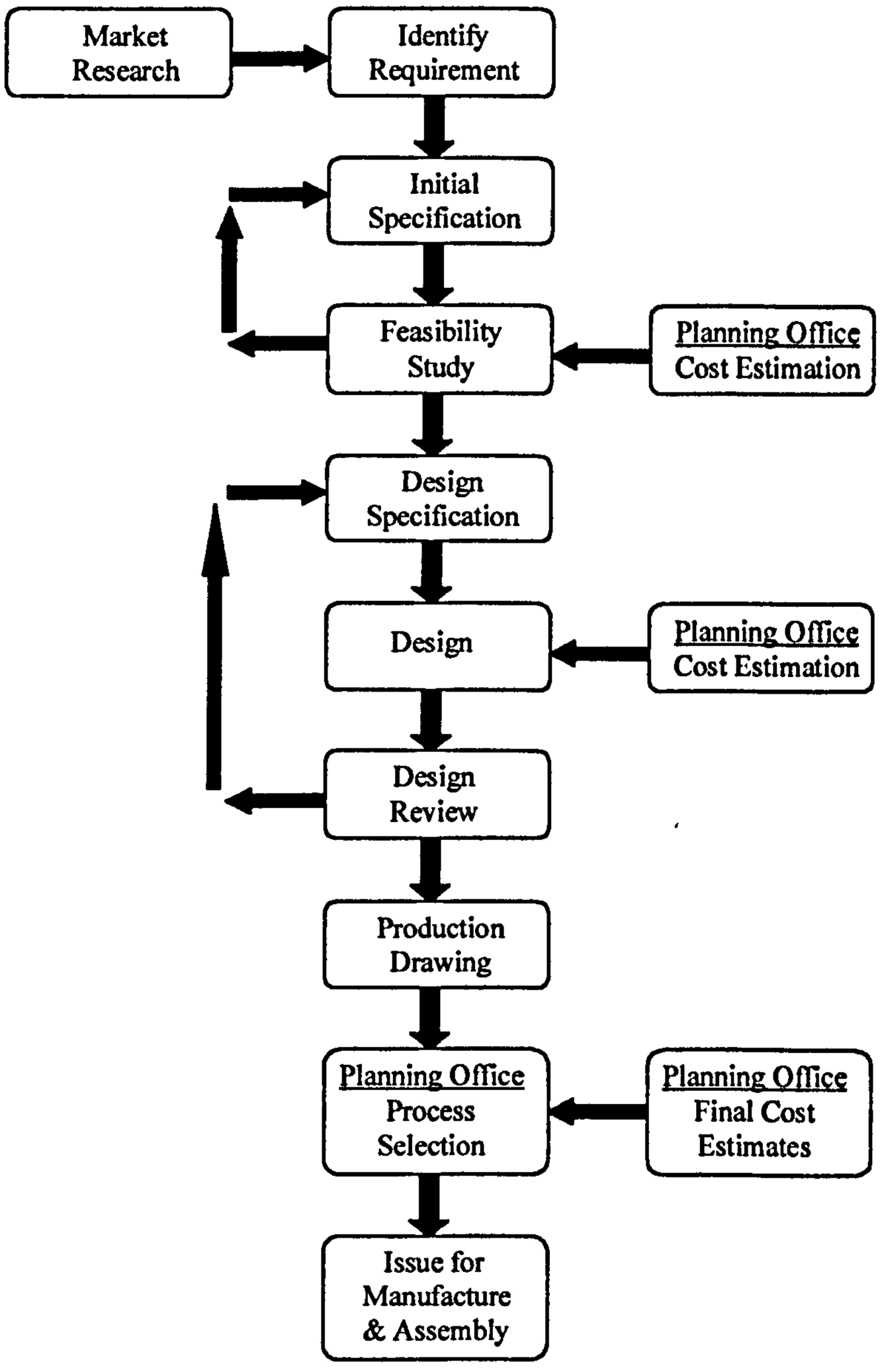


Figure a-1. Design Activity at CMB

Summarising the process in Figure a-1; the designer given a design specification looks at various design alternatives and considers possible trade-offs that may be associated with each alternative and selects a design path to follow. The detail design and the design review phases are iterative. Once the detail design has been finalised, planning advice is sought. In light of the planning advice, the design is revised and production drawings are made.

As the production drawings are completed, they are sent to the planning office where it is decided whether the component will be manufactured in-house or bought-in. The Make vs. Buy criteria applied at CMB are as follows:

- Capacity, specification and condition of factory equipment
- Simplicity or complexity of component design
- Competitive sub-contract rates for simple turned, milled or drilled components
- Ratio of "in-house" to sub-contract parts in assembly
- Ratio of "in-house" to sub-contract process operations
- Experience and capabilities of specialist manufacturers
- How the above parameters affect the handling and control of dependent associated components
- Availability of fixtures and special tools and manufacturing cell criteria.

Most of the machining centres on the shop floor are computer-controlled and have varying capacities. Therefore, CMB's manufacturing capabilities are geared towards manufacturing complex components of various sizes. The manufactured machines are hand-assembled at CMB.

If the component is to be manufactured in-house, then the planning office selects the relevant manufacturing process (or routing) that the component will follow on the shop floor. The final cost estimates are made at this stage. Based on the routing, the cost structure is built up automatically in the Integrated Business System in accounts. The cost structure primarily consists of:

- Cost for the raw material (accurate costs are not known at this stage, cost estimates are based on costs for similar materials bought in the past)
- In-house manufacturing operations
- Sub-contract costs

Each machining centre on the shop floor is treated as a cost centre in accounting and assigned a machine hourly rate using the cost absorption method. The standard cost of production is calculated as follows:

Standard Cost = Material Costs + (Set-up time x Machine Hourly Rate) +

(Unit time x batch size x Machine Hourly Rate) + Sub-contract costs.

Finally, the production drawings are issued for manufacture and assembly.

A2 Knowledge Elicitation

A2.1 Manufacturing Knowledge Elicitation

Source: One-to-one structured interviews with recognised domain expert (production planning manager).

Summary of Interviews

Most components manufactured on the machine floor have features that require drilling, turning or milling with a mixture of some surface and cylindrical grinding. These are the basic machine processes that need to be addressed first. Other less commonly used processes are slotting, honing, lapping etc. The machining criteria are summarised in section A. The work centre selection criteria is summarised in section B. Sections C, D and E define batch time, set-up time and unit time respectively. Section C also states the various contributions to the total component manufacturing cost. Section F gives a summary of sub-contract operations.

A. Machining Criteria

Surface Finish
Turning and Milling – capable of surface finishes down to 0.8 μm.
Below 0.8 μm, parts would normally be cylindrical ground down to 0.1 μm.
Below 0.1 μm parts would normally be polished, tapped or surface finished.

Tolerances
Turning and Milling – tolerances down to 0.02mm.
Below 0.02mm parts would normally be ground.
Position tolerances vary with size and specification of individual machining centres but are generally 0.025mm on hole centres.
Accuracy of drilled holes – nominal size to +0.1mm. Below +0.1mm, holes would be bored or bored and reamed to size.

Geometry
Turning concentricity when reversing for second operation – down to 0.05mm.
Below 0.05mm parts would be cylindrical ground.
Turning roundness and parallelism down to 0.01mm.
Flatness when milling – down to 0.03mm per 100mm length.
Below 0.03mm parts would be surface ground.
Squareness when milling – down to 0.03 mm.
Below 0.03mm parts would be surface ground.

B. Work Centre Selection

Estimates should be based on machining the component on the most cost effective work centre. This usually means selecting the smallest and most rigid machine that is capable of producing the features required to be machined. Generally, the smaller the work

centre, the lower the machine hourly rate and hence the lower machining cost. However, there are a number of other factors which may affect work centre selection such as machine accuracy, spindle orientation, machine specification, tool changer capacity, control specification on chuck size, machine rigidity etc. Wherever possible components should be machined on the CNC work centres for speed, accuracy and cost effectiveness. Selection of work centres may also be influenced to some degree by the demand and frequency of manufacture of particular components, and the need to machine other features on a specific work centre in a particular manufacturing cell. The work centres given in Table 10.1 should be the prime selection for machining components "in-house". The ideal component size or maximum desired component size is not the full machining envelope of the work centre, but a guide to the maximum size of component for estimating or costing purposes. Table 10.2 shows the work centres for secondary selection where capacity or component demand obviates choice from the primary selection.

C. Batch Time and Cost Calculations

$$Batch Time\ per\ Operation = Set-up\ time + (Unit\ time \times Batch\ quantity)$$

$$Total\ Operation Time\ per\ Component = Unit\ time + \frac{Set-up\ time}{Batch\ quantity}$$

$$Total\ Component\ Manufacturing\ Cost =$$

$$\sum_{mach=1}^{m} (Machine\ hourly\ rate \times (\frac{Set-up\ time}{Batch\ quantity} + \sum_{op=1}^{n} (Operation\ unit\ time)))$$

$$Sub-contract\ operation\ costs + Material\ Costs +$$

$$In-house\ process\ costs.$$

The Standard batch quantity used by the factory costing procedure is 3.

D. Set-up Times

Set-up time is allowed once per batch of components.
Machine set-up time consists of the following three main elements:

- Tool setting. This element relates directly to the number of tools used in the work centre operation. It includes obtaining or receiving the tools from the tool setting area or stores, checking the tools or performing any tool adjustments.
- Job preparation. This element includes receiving instructions including drawings and routings, studying the operational instructions, inspecting the "first-off" at the machine or at the inspection area, cleaning or stripping down the machine at the end of the machine cycle and loading or removing tools etc. from the machine.
- Machine setting. This element includes the fitting and setting up of special fixtures or other work holding devices such as vices, chucks, sine tables etc. Also includes input of tool data, offsets etc.

E. Unit Times

Unit (labour) time is allowed for every component in the batch. Unit time includes the following:

- Actual machining time (metal cutting),
- Machine positioning time,
- Component loading time,
- Tool change time,
- Any work alignments,
- Probing time for component orientation etc.,
- Local machining centre cleaning for work loading,
- Any interim releasing of work holding to relieve stresses induced during machining,
- In-process gauging where necessary, and,
- Any standard relaxation and contingency allowances.

F. Sub-contract Operations

Certain operations are sub-contracted and not completed "in-house". These are mainly the following:

- Machining of components larger than the capacity available on machine work centres in the manufacturing cells.
- Where components have features that demand accuracy greater than the accuracy specification of in-house equipment.
- Certain process operations such as chromium plating, nickel plating, hard anodising, ceramic coating, welding, brazing etc.
- Certain specialised machining operations such as gear cutting, splines and serrations(hobbing etc.), jig grinding, thread grinding, spark erosion, flame cutting/profile burning, bending, cam milling, fine polishing, engraving etc.
- All heat treatment operations such as annealing, stress relieving, hardening, tempering, vacuum work, tuftriding, nitriding, phosphate coating etc.

What follows are the task scripts elicited for three typical component parts that CMB manufacture in-house.

Task Script 1: Cost Estimate for the Seaming-Foll-Housing Component

Looking at the seaming-foll-housing component which is made from an aluminium block. We would normally machine a component like this on a machining centre because there are a number of manufacturing features on it, namely, drilling, tapping, boring and profile milling. There are some soft features on it as well as some angular features and there are holes in various different places. Pieces of this size we would normally machine on the Heckler & Koch or the Beaver machining centres, which have similar machine hourly rates. In this particular case because of it's depth we would normally put it on the Heckler & Koch and hold this part in a vice. On this particular part we've got a pre-machining operation on a vertical mill to machine off the two faces to hold it in a vice. As far as these estimates are concerned, we could work on doing the whole thing on the Beaver, which would be an extra setting but all the machining content should work out at a similar costing anyway. The criteria are a little bit complex for separating off pre-machining to offer a surface finish to start holding it in a machining centre.

First of all, this block has to be milled with a face mill. We've got to machine up the faces to hold it in a vice. We have to machine either the top face or the bottom face and the two side faces. As far as milling goes figures have been put together on face milling to get some crude answer to how all these work out. For a range of face widths, beginning with a face width of up to 50mm going up to 1000mm, the machining times for the first 50mm height are given. The time calculation for each additional 50mm on height is also given.

Step 1. Face mill the top face. The part is 124mm wide, so we're looking in the range of up to 150mm wide which gives us a time of 5mins for the first 50mm. The height of this face is (125+20 = 145) mm. Now we have an additional height of (145-50=95) mm to account for. The rule is that for each additional 50mm on height, add the respective number of minutes. Therefore, to face mill the top face takes a unit time of 5min+3min=8mins.

Step 2. Face mill the two sides. Each side has a face width of 124mm and a height of 82mm, however we've got an extra piece on it. We've made the block deeper so that we can hold it in a vice and machine the profile around and take the stock off the back. We've made it out of a block which is 110mm deep and so we have an extra (110-50 = 60)mm to account for. Therefore, to face mill the two sides takes a unit time of 2(5min+3min) = 16min.

Step 3. We've now got to profile around the shape. Profile round with an end mill around the shape and we'll probably have to go down in two depths because of the depth of the component. If you try and do it in one, the cutter would push off and cut the side walls taper. The figures for edge milling allow two cuts, because of the amount of stock coming off here we'll probably have to allow either extra cuts going around it or extra depths, one or the other. Here we'll work on extra depths and if necessary the estimates will have to be qualified for extra deep components in the future. We profile around then with an end mill. Refer to the table on edge milling. The rule is allow 25mm per minute + 2mins. So we need to know roughly the length of the periphery around the part. The periphery is approximately (145x2+124x2=538) mm. The actual value is probably more than that because it is going at an angle in certain places, so lets make it

550mm all around the periphery. So we have 550mm @ 25mm per min. Therefore, unit time to edge mill around the profile once is 22min+2min = 24min. Going around twice because it is deep gives a unit time of 2x24=48min.

Step 4. Machine for the slot feature (refer to slot milling). The rule is to allow 10mm per minute plus two minutes. We've got to run down the length of the slot, across the width then back up the slot again. So the length to go around is approx. 35mm twice plus 10mm = 80mm @ 10mm per min. Therefore, unit time to mill the slot once is 8min+2min=10min. Going around it twice because it is deep gives a unit time of 20min.

Step 5. Boring the two Ø30mm holes. These fall in the up to 50mm range, which gives 11mins each for drilling and boring to size. We need to account for the depth. The rule is that for holes greater than one diameter deep, add a further 50% for each additional diameter depth after the first diameter. So we have ((82-30)/30)=1.7 diameters extra on depth to account for. Therefore boring the two holes gives a unit time of 2(11x1.85)=40.7mins. This seems quite a lot of time.

Step 6. Boring the Ø70 centre bore. Let's say that this operation takes out just the 27mm depth. This gives a unit time of 11mins.

Step 7. Boring the Ø50mm centre bore. Let's say this operation takes out just the 27mm depth. This gives a unit time of 11mins.

Step 8. Drilling the Ø26mm centre hole. This is just drilled because there is no sizing tolerance specified for this hole. This gives a unit time of 4.5mins. We have other holes as well to account for.

Step 9. We have four holes that are drilled and counter-bored. Refer to figures on reaming. The drilled holes are Ø9mm, which give 2mins each. We have to account for the depth. The rule is that for holes greater than 3 diameters deep, add a further 20% for each additional diameter on depth. Depth is 82mm. So we have (82-27)/9=6 additional diameters to account for. Therefore to drill & counter-bore four holes takes a unit time of 4(2x2.2)=17.6mins.

Step 10. We have to drill the two Ø6.6mm holes which are 8mm deep. This gives a unit time of 2(1.5mins)=3mins.

Step 11. We have two tapped holes, Ø8.8mm and Ø6.8mm respectively, 10mm deep in another face. This gives a unit time of 2(2mins)=4mins.

Step 12. We have to drill an access hole Ø10mm. This gives a unit time of 1.5mins.

Step 13. Slotting. 42mm length slot. Rule is 5mins per 25mm length plus 5mins, which equals 15mins.

Step 14. We finally have to machine off that extra stock off the back. Refer to face milling. The figure for milling allows two cuts. A statement is required to say how much time is allowed for extra stock. Here, we start off at a 110mm, the figures allow for 3mm stock. So we have (110-3-82=25) mm extra stock. So we're looking at 25/3 = 8 more cuts across. From step 1, face milling the top face takes 7mins. Therefore, unit time = 8 cuts x 7mins = 56mins.

Total unit time =240mins. Factor for Aluminium gives 240 x .67 =160mins = 2.68hours.

In this case these elements have worked out quite well, where it might fall down is where we've got extremes like a particularly long feature. The standards therefore remain to be proved over time for other components.

Task Script 2: Cost Estimate for the Seaming-Turret Component

First of all we are looking at buying a blank in stainless steel. We are looking at buying what is called a plasma-cut or a flame-cut blank, which is a flame-cut disc of steel cut from a large plate with an allowance on the diameter, the usual diameter is something like 5 or 6mm a plate on something like this. Also whatever plate stock is nearest to the finished thickness with a few millimetres each side machining allowance is selected. A table can be produced of different plate sizes, which are available, which we could use for carbon steel or stainless steel. What we've selected here is a 63mm thick plate. We don't carry stocks of plates like this, we would buy this in as a flame-cut blank from a supplier; a list is available of carbon and steel plate thickness that our suppliers carry in stock which we could use as a guide. In this particular case we're calling for the part to be flame-cut from a 63mm thick plate, which leaves 3mm of stock on each face. The component finishes at 57mm, so we've got 3mm to face on both sides, bear in mind there is a little boss here, so it does give us extra stock on the face. Anyway, the plate is reasonably flat when purchased. So, the first thing to do is look at turning the outside diameter, producing the bore and facing each side. Essentially it is a turning operation to produce the diameter and the faces. It is easier to put the part in a chuck then to try and hold it on another way on a machining centre. However, it is a machining centre part for putting all the holes in the faces and the holes around the diameters. But just to produce the shape of the blank itself is really a turning operation. We are going to look at this first.

Step 1. First of all, look at the outside diameter, quite a big diameter. Taking it to be in the 300-500 mm diameter range, we have 8.75mins for turning a diameter, which is a maximum 25mm long. Turn O/D, we've got 8.75mins for turning the first 25mms, now the part is 63mm long i.e. 63-25=38mm extra length. The rule is for each additional 50mm length, add 33%. We're starting off with a diameter of 590mm and finishing at 586mms, therefore the figure given should be all right, since these figures are for the first 10mm of stock. Therefore, unit time to turn O/D is 8.75 x 1.33 =11.6mins.

Then we're looking at facing.

Step 2. Turning Face One. Face diameter = 586mm. Extrapolating from the 300mm range, we have 25mins for the first 3mm stock. We have 3mm each side, well this side has got 3mm on it anyway, so we use that. Unit time to face side one = 25mins.

Step 3. Turning Face Two. To face the other side we've got more than 3mm of stock because we've got a small boss here, which means we've got more to face off on the other side; i.e. 63-3-52=8mm extra stock. We've got the first cut of 25mins plus two more cuts 25+2(1.5x25)=100mins.

We can also do the centre bores here.

Step 4. Boring the centre bores:

For the Ø115mm, we have a unit time of 20mins;

For the Ø160mm, we have a unit time of 23mins; and,

For the Ø124mm, we have a unit time of 20mins.

Step 5. Machining the four chamfers gives a unit time of 2mins.

Total = 202mins. Factor for stainless steel = 202 x 1.5 = 303mins.

These are all based on NC times. The problem with this is that it won't be done on a NC machine, because it is too big. So, it would be done on a conventional machine. In most cases the ideal way would be the machining centre way but when the part comes outside the envelope of the machining centres, we have to multiply it by separate factors

for milling and turning. These factors will vary depending on the types of features. Regarding facing, once the operator has set the machine to face the feed rate would be very similar between a CNC machine and a conventional machine. The only gain that you would have between the two is that a CNC machine would be able to lock into constant surface speed per minute. So when you're facing down a face like this big disc here, the spindle reps. will speed up as it gets nearer to the middle. This is more efficient since you get a better finish because it'll be cutting with the same rate. Note that what we may need to do is set different factor rates regarding simple turned work, medium and complex turned work. Here the factor is taken to be 2 for a conventional machine, which gives 2 x 303 = 606mins = 10.1 hours.

We now need to look at the boring. There are a lot of holes in this part so the individual elements of the standards will be tested here. So really we're looking at drilling and boring now. This part is to be machined on the PMC4, which is not a full CNC machine, but at the moment we would plan it for that machine because it is easier to produce a program for this machine than for a CNC MACHINE. The PMC4 is an NC horizontal borer which is not capable of contouring, which is not needed here, but it allows you to program the positions and program the depths so it's sort of halfway to a full blown machining centre. So the best thing to do is to plug our way through all these holes and just use the drilling and boring elements. Starting with the bores:

Step 1. Ø84mm bore x 12 @ 17mins each gives a unit time of 204mins. These are counter-bored. Note: The boring elements contain an allowance for drilling, so for counter-bores, we've already drilled them. So, another band needs to be added, the rule being that for counter-bores we only need to add 50% of the original time so that the drilling element is not duplicated.

Step 2. Ø90mm counter-bore x 12 @ 8.5mins each gives a unit time of 102mins.

Step 3. Ø30mm bores x 24 @ 11mins each gives a unit time of 264mins. These are counter-bored as well.

Step 4. Ø52mm counter-bores x 24 @ 5.5mins each gives a unit time of 132mins.

Step 5. Drilled and tapped holes, we've got a cluster of three around each of the 24 holes and they are M6 taps. So we've got a unit time of 72 x 2 = 144mins.

Step 6. 6 holes, drilled and tapped, M20 @ 6mins each gives a unit time of 36mins.

Step 7. 4 holes, drilled and tapped, M8 @ 2mins each gives a unit time of 8mins.

Step 8. 6 holes, drilled and tapped, M16 @ 4mins each gives a unit time of 24mins.

Step 9. Ø8.7mm x 24 holes, drilled and spot-faced, in the periphery @ 2mins each gives a unit time of 48mins.

Step 10. 2 holes Ø8mm, 195mm deep, drilled, counter-bored, tapped and spot-faced. So use the drilled and spot-faced values. It is greater than 3 diameters deep, so we need to add a further 20% for each additional diameter. It is (195-(8 x 3))/8=22 extra diameters on depth to account for. So unit time = 2(2 x 5.4) = 21.6mins.

Total = 983.6mins.

Factor for stainless steel = 1.5 x 983.6 = 1475mins = 24.59 hours.

Task Script 3: Cost Estimate for the Spring-Plunger Component

This is a steel part with round features. To commence machining this we need to put it on a NC lathe because we are looking at the most efficient way of turning this part. We would normally put this part on the Tsugami or the Index lathe. In this instance the index lathe has been selected. Basically, we're looking at turning the diameters and boring the holes out. Also we have a note that this part is made of EN34 steel and it's got to be carburised and hardened on certain features, having said that we need to look at the features to see where the part is hardened and the hardening is indicated by an asterisk. From the drawing we can see that this applies to one of the bores and the larger external diameter. So the main part left soft is the thread, the external thread so what we are looking at is turning this part plus a grinding allowance. The grinding allowance is required because the part will probably distort in hardening. Sizes cannot be guaranteed after heat treatment. So we turn this part leaving an allowance in the bore and an allowance on the outside diameter to grind at a later stage after the hardening has been carried out. So to begin with then we look at the stock size that we have in store to see whether the part can be made out of stock size EN34bar. From the stock list we see that the part can be made out of 60mm diameter bar which is a size we carry in stock. Therefore the first thing we do is look at the material specification to see whether we carry the stocks of that material bearing in mind that the designers should primarily look at the stock list and design parts out of material on the stock list if at all possible. Where the nature of the work demands something else we would have to make it out of material that is not on the stock list in which case we may have to buy the material in just for that particular job and not carry it as a stock item. So in this case we can look at the EN34 on the stock list which is nickel chrome casehardened steel and the diameters we carry are in the range of 25-200mm. We've picked out the 60mm diameter because the outside external diameter finishes at approx. 54mm, so that's the nearest stock size to select. This means that on the largest diameter then there is 6mm stock to turn off, bearing in mind that we're going to turn it leaving a grinding allowance on it. We then have to decide on the length as well. This part could be made as singles or we could make it in a groove bar. As it happened with this we decided to make it in singles. So we select a piece of material and put an allowance on so that we can hold it in a chuck and then we put a basic allowance on to clean the faces. This particular part we're turning is round anyway so we don't need to allow stock on that. The overall length finishes at 80mm; we've actually cut it off at 90mm long so there is 5mm a side to machine off at some stage. How much you need for holding on to a part depends on the nature of the work, it varies from down to 5mm, most likely less on some particular components. The rule is that if a part is a fairly big heavy part which projects from the chuck a long way then we need a longer length to hold on because of the weight ratio trying to pull it out of the jaw when you're machining it. So on a lighter part you need less to hold on to. In this particular case we've allowed 10mm which is a facing allowance plus something to hold on to. This allows us to turn off the two diameters and when the features have been turned on one side we would turn it around and hold it on the other side and face the remaining side to produce the counter-bore. So that's the material selected.

The next thing to do then is to decide what is the first operation, which would in this case be turning. The features that are to be hardened would be turned leaving a grinding allowance so that they can be ground to the particular tolerances after

hardening. The remaining features we would actually turn to size, which are the thread and the grooves at this stage and the keyways to be put in at a later stage on another work centre.

Operation Turning

The first thing to do is to hold it in a chuck on the lathe and to remove all the extra stock from the diameters and the face and to drill and bore the centre hole to whatever size we think we need at this stage. We have a complication in this in that we have an internal keyway to broach, which means we need to mount the component on a broach adapter to support the component while the keyway is machined out. So as well as leaving a grinding allowance we're going to turn this bore to a size which will enable a precision fit onto the broaching adapter. In this case we know that we have a broach adapter of 25.4mm so we need to leave some stock in to produce a fit on to the broach adapter. Other than that we 're just turning the part up to leave a grinding allowance on the diameter that will be ground after heat treatment. So the first thing to do is to turn the outside diameter, it doesn't matter which order we take it in because we're removing the same amount of stock from the part whichever route we follow.

The figures used are based on steel and cast iron and proportioned up or down for other materials such as stainless steel which takes longer to machine or aluminium/brass which takes less to machine. The figures on turning are based on turning to coarse limits which is down to .05mm on the diameter or to include grinding allowances which brings it into this category.

Step 1. Turning the larger outside diameter. We're going to turn from the end including the thread. The length of the part is 80mm; the diameter starts at 60mm, we're cutting down to 55mm. So the process falls into the range of diameter 50-75mm which gives a unit time of 1.25mins for the first 25mm length. However for each extra 50mm on length we want to add another 33% on there. Therefore, 80-25=55 and for every 50mm extra we're adding 33% on, so it falls into two portions. Therefore, turning the outside diameter takes a unit time of (1.25 x 1.33 x 1.33)=2.21mins.

Step 2. Turning the thread (refer to threading table). We've got a 50mm thread, which gives us 1.2mins for the first 25mms of length, for each additional 25mm we need to add 50% on. The thread is 40mm long i.e. we have 15mm extra length. Adding 50%, turning the thread takes a unit time of (1.2 x 1.5)=1.8mins.

Step 3. Turning Face One. The end face starts at 60mm diameter of stock so we're looking at the range 50-75mm, so we've got 0.45mins for that and then for each additional 3mm of stock we add 50% on. So, we've got altogether 10mm of stock on there, so we've got to take 5mm off each side. So we just need to add an extra 50%. Therefore, turning one end face takes a unit time of (0.45 x 1.5)= 0.675mins.

Step 4. Drill and ream hole to get to the right size. We've got to drill it down the middle. Drill to the full length of it. Diameter of hole is approx. 26mm, so we're in the range 20-30mm that gives a unit time of 6mins. The depth that we have to drill to is 80mm. That is just over 3 diameters to go through. For each additional diameter we need to add 20% on i.e. unit time = (6 x 1.2)=7.2mins.

Step 5. Turn counter-bore by facing. The counter-bore is treated as a recess. Diameter is 42mm, which is in the range 25-50mm, which gives a unit time of 0.15mins. Depth to turn is 23mm. For each additional 3mm stock after the first 3mm add 50%. We have (23-3=20)/3≅7 extra cuts, i.e. unit time = (0.15+0.15(0.5x7))=0.675mins.

Step 6. Turn the four grooves @ 1min each gives a unit time of 4mins.

Step 7. Turn the two undercuts @ 1min each gives a unit time of 2mins.
Step 8. Turn the four chamfers @ 0.5mins each gives a unit time of 2mins.
Step 9. Face the remaining end. Unit time = (0.45 x 1.5)=0.675mins.
Total = 21mins.

Operation Broaching
Simple standards for broaching which should hold good because it is the same criteria we use for putting in the machining time. The standard allowance is 5mins for the first pull of the broach. If the keyway is excessively deep then we have to re-pull, therefore allow 3mins for each additional pull. Allowing two pulls gives a unit time of **8mins.**

Operation Milling
The external keyway has to be milled. The keyway is 8mm wide, 4mm deep so it falls within the first bracket. The length of the keyway is 30mm long. 10mm per minute gives a unit time of **5mins.**

Operation Grinding
Step 1. External grind the outside diameter which is 54mm nominal diameter. The grinding is split into two tolerance bands. This particular tolerance here is .05, so that falls into the coarse band. For each diameter plunge ground and swept gives a unit time of 8mins. Qualifying this, we're saying that, for plunge grinding we're looking at a length of up to a maximum of 30mm, above that we need to apply a factor. The length of the diameter is 40mm, so we have that grinding the outside diameter takes a unit time of (8 x 1.33)=10.6mins.
Step 2. Internal grind the bore. The tolerance on the bore is .03 so it's in the coarse band. Each diameter plunge ground takes a unit time of 10mins. The bore is (80-23)=57mm. The rule is that for each additional 25mm over the first 25mm we add 33%, so 57-25=32, i.e. two more lengths to take account of. Therefore, to internal grind the bore takes a unit time of (10 x 1.33 x 1.33)=17.7mins.
Step 3. Kiss grind the counter-bore face to size. Allowance for kissing faces which gives time for cleaning the face up, to measure it and then go in a second time is allocated a unit time of 2mins.
Total = **30mins.**

A2.2 Assembly Knowledge Elicitation

Source: *Boothroyd G., Dewhurst P. and Knight W. (1994).*

RULESET (Component Coding Rules for Manual Assembly)

Rule 1:
If (parts can be grasped and manipulated by one hand without the aid of grasping tools)
If ((α+β) < 360°)
Then 1st digit of part handling code = 0;
If (360° ≤ (α+B) < 540°)
Then 1st digit of part handling code = 1;
If (540° ≤ (α+B) < 720°)
Then 1st digit of part handling code = 2;
If ((α+B) = 720°)
Then 1st digit of part handling code = 3;

Rule 2:
If (parts can be grasped and manipulated by one hand but only with the aid of grasping tools)
If (α ≤ 180° and 0 ≤ β ≤ 180°)
Then 1st digit of part handling code = 4;
If (α ≤ 180° and β = 360°)
Then 1st digit of part handling code = 5;
If (α = 360° and 0 ≤ β ≤ 180°)
Then 1st digit of part handling code = 6;
If (α = 360° and β = 360°)
Then 1st digit of part handling code =7;

Rule 3:
If (two hands are required for manipulation if parts severely nest or tangle or are flexible but can be grasped and lifted by one hand with the use of grasping tools if necessary)
Then 1st digit of part handling code = 8;

Rule 4:
If (two hands are required for large size or two persons are required or mechanical assistance required for grasping and transporting parts)
Then 1st digit of part handling code = 9;

Rule 5:
If (1st digit of part handling code = 0,1,2 or 3)
If (parts are easy to grasp and manipulate)
If (thickness > 2mm and size > 15mm)
Then 2nd digit of part handling code = 0;
If (thickness > 2mm and 6mm ≤ size ≤ 15mm)
Then 2nd digit of part handling code = 1;
If (thickness >2mm and size < 6mm)
Then 2nd digit of part handling code = 2;
If (thickness ≤ 2mm and size > 6mm)

Then 2nd digit of part handling code = 3;
If (thickness ≤ 2mm and size ≤ 6mm)
Then 2nd digit of part handling code = 4;
Else
If (thickness > 2mm and size > 15mm)
Then 2nd digit of part handling code = 5;
If (thickness > 2mm and 6mm ≤ size ≤ 15mm)
Then 2nd digit of part handling code = 6;
If (thickness > 2mm and size < 6mm)
Then 2nd digit of part handling code = 7;
If (thickness ≤ 2mm and size > 6mm)
Then 2nd digit of part handling code = 8;
If (thickness ≤ 2mm and size ≤ 6mm)
Then 2nd digit of part handling code = 9;

Rule 6:

If (1st digit of part handling code = 4,5,6 or 7)
If (parts need tweezers for grasping and manipulation)
If (parts can be manipulated without optical magnification)
If (parts are easy to grasp and manipulate)
If (thickness > 0.25mm)
Then 2nd digit of part handling code = 0;
If (thickness ≤ 0.25mm)
Then 2nd digit of part handling code = 1;
Else
If (thickness > 0.25mm)
Then 2nd digit of part handling code = 2;
If (thickness ≤ 0.25mm)
Then 2nd digit of part handling code = 3;
If (parts require optical magnification for manipulation)
If (parts are easy to grasp and manipulate)
If (thickness > 0.25mm)
Then 2nd digit of part handling code = 4;
If (thickness ≤ 0.25mm)
Then 2nd digit of part handling code = 5;
Else
If (thickness > 0.25mm)
Then 2nd digit of part handling code = 6;
If (thickness ≤ 0.25mm)
Then 2nd digit of part handling code = 7;
If (parts need standard tools other than tweezers)
Then 2nd digit of part handling code = 8;
If (parts need special tools for grasping and manipulation)
Then 2nd digit of part handling code = 9;

Rule 7:

If (1st digit of part handling code = 8)
If (parts present no additional handling difficulties)
If (α ≤ 180° and size > 15mm)

Then 2nd digit of part handling code = 0;
If ($\alpha \leq 180°$ and 6mm ≤ size ≤ 15mm)
Then 2nd digit of part handling code = 1;
If ($\alpha \leq 180°$ and size < 6mm)
Then 2nd digit of part handling code = 2;
If ($\alpha = 360°$ and size > 6mm)
Then 2nd digit of part handling code = 3;
If ($\alpha = 360°$ and size ≤ 6mm)
Then 2nd digit of part handling code = 4;

If (parts present additional handling difficulties, e.g. sticky, delicate, slippery, etc)

If ($\alpha \leq 180°$ and size > 15mm)
Then 2nd digit of part handling code = 5;
If ($\alpha \leq 180°$ and 6mm ≤ size ≤ 15mm)
Then 2nd digit of part handling code = 6;
If ($\alpha \leq 180°$ and size < 6mm)
Then 2nd digit of part handling code = 7;
If ($\alpha = 360°$ and size > 6mm)
Then 2nd digit of part handling code = 8;
If ($\alpha = 360°$ and size ≤ 6mm)
Then 2nd digit of part handling code = 9;

Rule 8:
If (1st digit of part handling code = 9)
If (parts can be handled by one person without mechanical assistance)
If (parts do not severely nest or tangle and are not flexible)
If (part weight < 10lb)
If (parts are easy to grasp and manipulate)
If ($\alpha \leq 180°$)
Then 2nd digit of part handling code = 0;
If ($\alpha = 360°$)
Then 2nd digit of part handling code = 1;
If (parts present other handling difficulties)
If ($\alpha \leq 180°$)
Then 2nd digit of part handling code = 2;
If ($\alpha = 360°$)
Then 2nd digit of part handling code = 3;
If (parts are heavy >10lb)
If (parts are easy to grasp and manipulate)
If ($\alpha \leq 180°$)
Then 2nd digit of part handling code = 4;
If ($\alpha = 360°$)
Then 2nd digit of part handling code = 5;
If (parts present other handling difficulties)
If ($\alpha \leq 180°$)
Then 2nd digit of part handling code = 6;

If ($\alpha = 360°$)
Then 2nd digit of part handling code = 7;
If (parts severely nest or tangle or are flexible)
Then 2nd digit of part handling code = 8;
If (two persons or mechanical assistance required for parts manipulation)
Then 2nd digit of part handling code = 9;

Rule 9:

If (part added but not secured)
If (part and associated tool (including hands) can easily reach the desired location)
Then 1st digit of part insertion code = 0;
If (part and associated tool cannot easily reach the desired location
due to obstructed access **or** restricted vision)
Then 1st digit of part insertion code = 1;
If (part and associated tool cannot easily reach the desired location
due to obstructed access **and** restricted vision)
Then 1st digit of part insertion code = 2;

Rule 10:

If (part added and secured immediately)
If (part and associated tool (including hands) can easily reach the desired location
and the tool can be operated easily)
Then 1st digit of part insertion code is 3;
If (part and associated tool (including hands) cannot easily reach the desired location
or the tool cannot be operated easily due to obstructed access **or** restricted vision)
Then 1st digit of part insertion code is 4;
If (part and associated tool (including hands) cannot easily reach the desired location
or the tool cannot be operated easily due to obstructed access **and** restricted vision)
Then 1st digit of part insertion code is 5;

Rule 11:

If (separate assembly operation where all solid parts are in place)
Then 1st digit of part insertion code is 9;

Rule 12:

If (1st digit of part insertion code is 0,1 or 2)
If (after assembly no holding down required to maintain orientation and location)
If (easy to align and position during assembly)
If (no resistance to insertion)
Then 2nd digit of insertion code = 0;
If (resistance to insertion)
Then 2nd digit of insertion code =1;
If (not easy to align or position during assembly)

If (no resistance to insertion)
Then 2nd digit of insertion code = 2;
If (resistance to insertion)
Then 2nd digit of insertion code =3;

If (holding down required during subsequent processes to maintain orientation or location)

If (easy to align and position during assembly)
If (no resistance to insertion)
Then 2nd digit of insertion code = 6;
If (resistance to insertion)
Then 2nd digit of insertion code =7;
If (not easy to align or position during assembly)
If (no resistance to insertion)
Then 2nd digit of insertion code = 8;
If (resistance to insertion)
Then 2nd digit of insertion code =9;

Rule 13:

If (1st digit of part insertion code is 3,4 or 5)

If (no screwing operation or plastic deformation immediately after insertion)
If (easy to align and position with no resistance to insertion)
Then 2nd digit of insertion code = 0;
If (not easy to align and position and/or resistance to insertion)
Then 2nd digit of insertion code = 1;

If (plastic deformation immediately after insertion)
If (plastic bending or torsion)
If (easy to align or position during assembly)
Then 2nd digit of insertion code = 2;
If (not easy to align or position during assembly)
If (no resistance to insertion)
Then 2nd digit of insertion code = 3;
If (resistance to insertion)
Then 2nd digit of insertion code = 4;
If (riveting or similar operation)
If (easy to align or position during assembly)
Then 2nd digit of insertion code = 5;
If (not easy to align or position during assembly)
If (no resistance to insertion)
Then 2nd digit of insertion code = 6;
If (resistance to insertion)
Then 2nd digit of insertion code = 7;

If (screw tightening immediately after insertion)
If (easy to align and position with no torsional resistance)
Then 2nd digit of insertion code = 8;
If (not easy to align and position and/or torsional resistance)
Then 2nd digit of insertion code = 9;

Rule 14:

If (1st digit of part insertion code is 9)

If (mechanical fastening processes for parts already in place but not secured immediately after insertion)

If (none or localised plastic deformation)

If (bending or similar processes)

Then 2nd digit of insertion code = 0;

If (riveting or similar processes)

Then 2nd digit of insertion code = 1;

If (screw tightening or other processes)

Then 2nd digit of insertion code is 2;

If (bulk plastic deformation i.e. large proportion of part is plastically deformed during fastening)

Then 2nd digit of insertion code is 3;

If (non-mechanical processes for parts already in place but not secured immediately after insertion)

If (metallurgical processes)

If (no additional material required e.g. resistance, friction welding etc.)

Then 2nd digit of insertion code = 4;

If (additional material required)

If (soldering processes)

Then 2nd digit of insertion code = 5;

If (weld/braze processes)

Then 2nd digit of insertion code = 6;

If (chemical processes e.g. adhesive bonding etc.)

Then 2nd digit of insertion code = 7;

If (non-fastening processes)

If (manipulation of parts or sub-assembly e.g. orienting, fitting etc.)

Then 2nd digit of insertion code = 8;

If (other processes e.g. liquid insertion etc.)

Then 2nd digit of insertion code = 9;

A3 Data Tables

Table 1. Average Set-up Times		
Work Centre	**Code**	**Average set-up time (min)**
Matchmaker	VMT1	30
Beaver	VMT2	45
Heckler & Koch	VMC2	45
Wadkin	HMC1	15
Mandelli 8	LMC1	30
Mandelli Regent	LMC2	30
Vertical Borer	VB	45
Chucking Lathes	CTS,CTM,CTL	30
Centre Lathes	CLS,CLM,CLL	45
Vertical Milling m/c's	VMM,VML	30
Horizontal Milling m/c's	HMM	30
Broaching	BR	30
Slotting	SL	30
Radial Drilling	RD	30
Sensitive Drilling	SD	15
Surface Grind	SGM,SGL	15
Ring Grind	RG	15
Universal Grind	UGM,UGL	45
External Grind	EGL,EGT	30
Naxos	NAX	45
Giddings & Lewis	JMTGL	60
Lapping	LAP	15
Honing	HON	15

Table 2.1. Unit Times for Milling Faces

Figures based on the following:

- Coarse limits (down to 0.05mm flatness)
- Allow for 3mm stock
- Two Cuts

Face Width (mm)	Unit Time (min)	For each additional 50mm Height (min)
50	2.5	+1
75	3.0	+1.5
100	4.0	+2
150	5.0	+3
200	6.0	+4
		For each additional 100mm Height
250	7.0	+5
300	8.0	+6
400	9.0	+7.5
500	10.0	+9.5
750	12.0	+14
1000	22.0	+19

Table 2.2. Unit Times for Milling Edges

Figures based on the following:

- Coarse limits (down to 0.05mm flatness)
- Allow for 3mm stock
- Two Cuts

Rule: Unit Time = 2 min + 1 min per 25mm periphery

Table 2.3. Unit Times for Milling Slots

Figures for keyways etc. based on the following:

- Depth up to 5mm (For every additional 5mm on depth, x 175%)
- Width up to 10mm (For every additional 10mm on width, x 175%)

Rule: Unit Time = 2 min + 1 min per 10mm of length

Table 3. Unit Times for Broaching

Rule: Unit Time = 5 min for first pull + 3 min each subsequent pull

Table 4. Unit Times for Slotting

Rule: Unit Time = 5 min + 5 min per 25mm length

Table 5. Unit Times for Boring

Figures based on the following:

- Holes up to 1 diameter deep (where depth is greater than 1 diameter, add a further 50% for each diameter)
- H8 limit (tolerances greater than H8 add 15%)

Diameter (mm)	Time (min)
≤15	5
15-25	8
25-50	11
50-75	14
75-100	17
100-150	20
150-200	23
200-300	26

Table 6. Unit Times for Drilling

Figures based on the following:

- Drilled holes up to 3 diameters deep (where depth is greater than 3 diameters, for each additional diameter on depth, add a further 20%)

Diameter (mm)	Time (min)
≤10	1.5
11-20	3.0
21-30	4.5
31-40	6.0
41-50	7.5
51-60	9.0
61-70	10.5
71-80	12.0

Table 7. Unit Times for Drilling Reamed holes, Tapped holes, Drilled & Spot-faced holes and Drilled & Counter-bored holes

Figures based on the following:

- Drilled holes up to 3 diameters deep (where depth is greater than 3 diameters, for each additional diameter on depth, add a further 20%)

Diameter (mm)	Time (min)
≤10	2
11-20	4
21-30	6
31-40	8
41-50	10
51-60	12
61-70	14
71-80	16

Table 8.1. Unit Times for Turning Faces

Figures based on the following:

- 2 cuts
- depth of stock up to 3mm (for each additional 3mm stock add 50% to the time)

Face Diameter (mm)	Time (min)
≤15	0.1
16-25	0.12
26-50	0.15
51-75	0.45
76-100	1.1
101-150	2.2
151-200	4.2
201-300	9.0
301-500	25.0
501-1000	105.0

Table 8.2. Unit Times for Turning Diameters

Figures based on the following:

- turning to coarse limits i.e. down to 0.05mm, or to grinding allowances (for fine limits i.e. down to 0.02mm, add a further 10%)
- for the first 25mm length (for every additional 50mm on length over the first 25mm length of diameter add 33%)
- times include 2 roughing cuts plus 1 finishing cut for up to 10mm stock on diameter (for each additional 10mm stock add 50%)

Diameter (mm)	Time (min)
≤15	0.25
16-25	0.45
26-50	0.9
51-75	1.25
76-100	1.75
101-150	2.6
151-200	3.5
201-300	5.25
301-500	8.75

Table 8.3. Unit Times for Turning Grooves
Rule: For Diameter grooves, undercuts etc. Unit Time = 1 min each For Facial grooves etc. Unit Time = 3 min each

Table 8.4. Unit Times for Turning Chamfer
Rule: For internal or external chamfers, Unit Time = 0.5 min each

Table 8.5. Unit Times for Turning Threads

Figures based on the following:

- first 25mm length (for each additional 25mm length add 50%)

Nominal Diameter of Thread (mm)	Time (min)
10	0.3
20	0.4
30	0.5
40	0.8
50	1.2
60	1.4
70	1.7
80	2.0

Table 9.1. Unit Times for External Grinding (cylindrical)

Figures based on the following:

- Diameters plunge ground and swept up to 30 mm length (for lengths greater than 30mm i.e. Traverse grinding, add 33% for every additional 50mm length over the first 30mm)
- Surface finishes above 0.4µm (for surface finishes below 0.4µm add 20%)

Rule: For tolerances greater than or equal to 0.01mm, Unit Time = 8 min else 10 min.

Table 9.2. Unit Times for Internal Grinding (cylindrical)

Figures based on the following:

- Diameters plunge ground and swept up to 25mm length (for lengths greater than 25mm i.e. traverse grinding, add 33% for every additional 50mm length over the first 30mm
- Surface Finishes above 0.4µm (for surface finishes below 0.4µm add 25%)

Rule: For tolerances greater than or equal to 0.01mm, Unit Time =10 min else 12 min.

Table 9.3. Unit Times for Surface Grinding

Figures based on the following:

- 0.25/0.3mm stock allowance
- Tolerances of 0.01 & above (for tolerances below 0.01 add a further 25%)
- First 25mm length (for each additional 25mm length add 1 minute)

Width (mm)	Time (min)
≤25	4
26-50	5
51-75	6
76-100	7
101-150	9
151-200	11
201-300	15
301-500	23

Table 9.4. Unit Times for 'Kiss Grinding' step faces using side of wheels

Figures based on the following:

- 0.05mm stock allowance.

Step Length (mm)	Time (min)
≤50	1
51-100	2
101-200	3
201-300	4
301-500	5

Table 10.1. Primary Selection Work Centres		
Work Centre	Code	Max. Desired Component Size
Tsugami	TMC1	Chuck work Ø250mm x 150mm Shaft work Ø100mm x 500mm (Note: chuck and shaft work selected for this work centre would normally have other machined features in faces or diameters such as holes, keyways, steps etc.)
Index	IND	Chuck work Ø300mm x 150mm Shaft work Ø80mm x 1000mm (Note: This machine is used for turning external diameters and faces as well as boring, drilling and tapping holes on the component centre axis only.)
Matchmaker	VMT1	200mm x 100mm x 10mm
Beaver	VMT2	300mm x 150mm x 20mm
Heckler & Koch	VMC2	700mm x 300mm x 150mm
Wadkin	HMC1	150mm x 150mm x 100mm
Mandelli 8	LMC2	800mm x 500mm x 250mm
Mandelli Regent	LMC1	1500mm x 1000mm x 800mm
Vertical Borer	VB	Ø1200mm x 600mm

Table 10.2. Secondary Selection Work Centres		
Work Centre	Code	Component Size
Chucking Lathes	CTS,CTM,CTL	Ø300mm x 150mm
Centre Lathes	CLS,CLM,CLL	Chuck work Ø500mm x150mm Shaft work Ø200mm x 2000mm
Vertical Milling	VMM,VML	700mm x 300mm x 100mm
Horizontal Milling	HMM	700mm x 300mm x 100mm
Broaching Keyways	BR	25 mm wide x 250mm
Slotting Keyways	SL	25mm wide x 200mm
Giddings & Lewis Bores	JMTGL	2000mm x 2000mm x 1000mm
Radial Drilling	RD	Up to Ø50mm holes for components of size 700mm x 600mm x 300mm
Sensitive Drilling	SD	Up to Ø12mm holes for components of size 200mm x 200mm x 25mm
Surface Grind	SGM,SGL	1200mm x 500mm x 150mm
Ring Grind	RG	Ø600mm x 250mm
Universal Grind	UGM,UGL	Ø300mm x 500mm
External Grind	EGL,EGT	Ø300mm x 1500mm
Naxos (Large External)	NAX	Ø450mm x 2000mm
Lapping	LAP	Ø200mm Face
Honing	HON	Up to Ø50mm holes.

[1] **Note. All the figures in these data tables are based on CNC times for Steel/Cast Iron. Factors for Stainless Steel and Aluminium/Brass are 150% and 67% respectively.**

Table 11.1. Lookup Table for Manual Handling Times. (s)

Code digit	0	1	2	3	4	5	6	7	8	9
0	1.13	1.43	1.88	1.69	2.18	1.84	2.17	2.65	2.45	2.98
1	1.5	1.8	2.25	2.06	2.55	2.25	2.57	3.06	3	3.38
2	1.8	2.1	2.55	2.36	2.85	2.57	2.9	3.38	3.18	3.7
3	1.95	2.25	2.7	2.51	3	2.73	3.06	3.55	3.34	4
4	3.6	6.85	4.35	7.6	5.6	8.35	6.35	8.6	7	7
5	4	7.25	4.75	8	6	8.75	6.75	9	8	8
6	4.8	8.05	5.55	8.8	6.8	9.55	7.55	9.8	8	9
7	5.1	8.35	5.85	9.1	7.1	9.55	7.85	10.1	9	10
8	4.1	4.5	5.1	5.6	6.75	5	5.25	5.85	6.35	7
9	2	3	2	3	3	4	4	5	7	9

Table 11.2. Lookup Table for Manual Insertion Times. (s)

Code digit	0	1	2	3	4	5	6	7	8	9
0	1.5	2.5	2.5	3.5	-	-	5.5	6.5	6.5	7.5
1	4	5	5	6	-	-	8	9	9	10
2	5.5	6.5	6.5	7.5	-	-	9.5	10.5	10.5	11.5
3	2	5	4	5	6	7	8	9	6	8
4	4.5	7.5	6.5	7.5	8.5	9.5	10.5	11.5	8.5	10.5
5	6	9	8	9	10	11	12	13	10	12
9	4	7	5	3.5	7	8	12	12	9	12

A4 Design Examples

What follows are the detail designs of the representative components referred to for illustrative examples of the designed methodology.

2001381 C

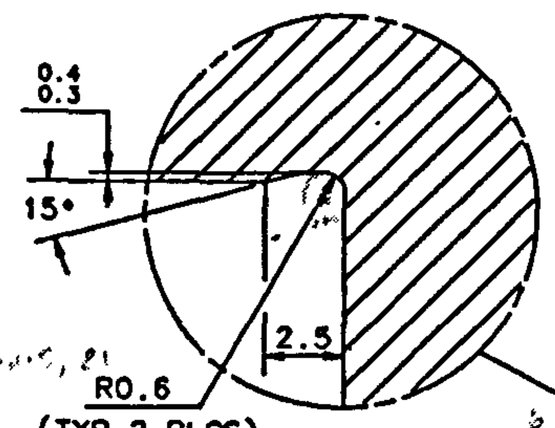
Sheet 1 of 1 Sheets

PROJECTION

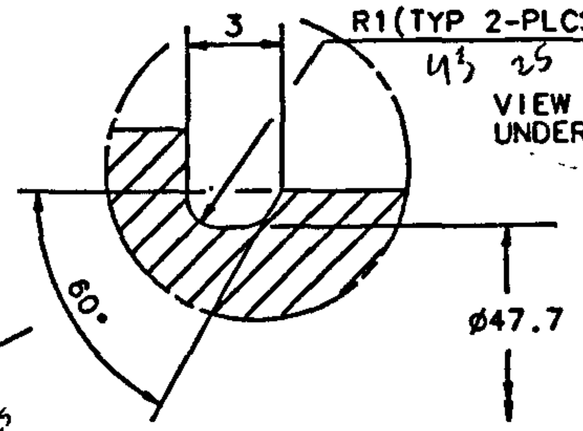
DO NOT SCALE

IF IN DOUBT-ASK

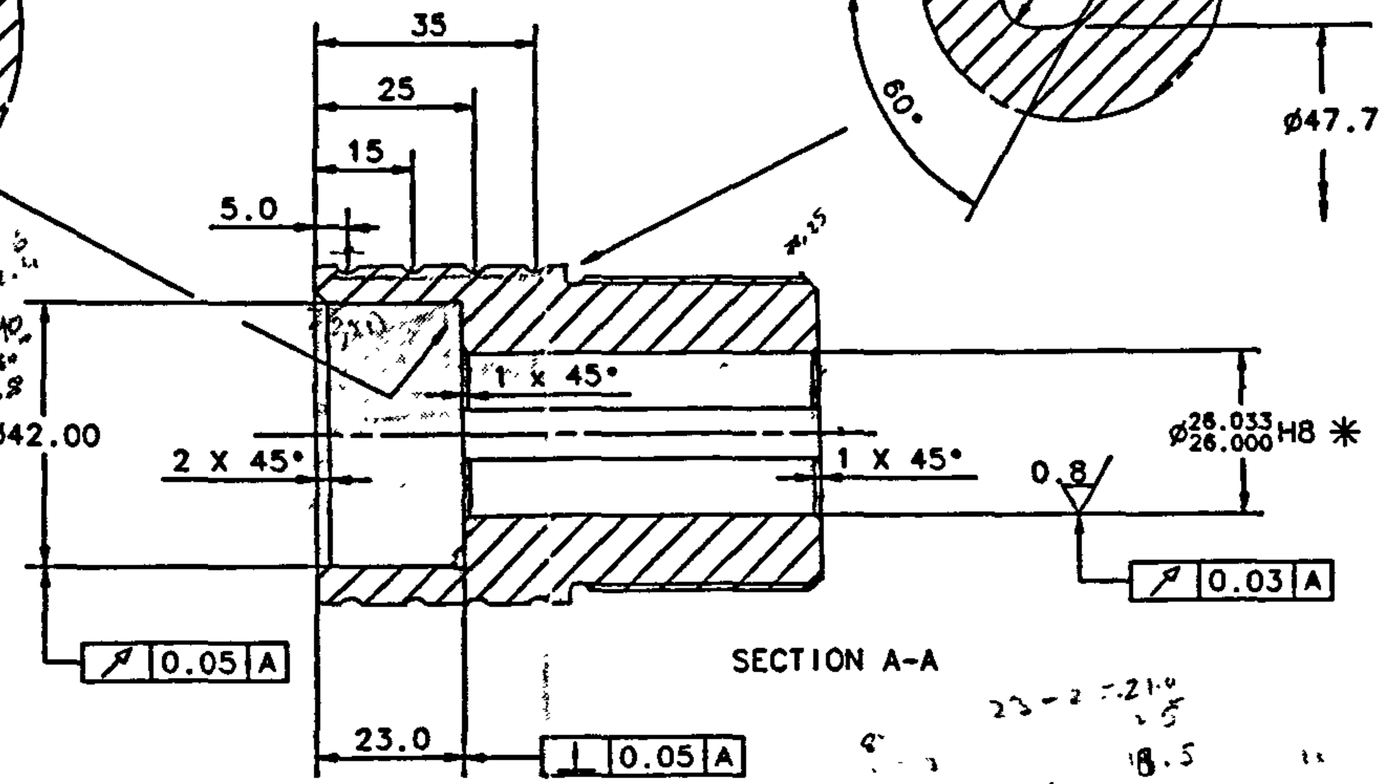
DIMENSIONS IN MILLIMETRES UNLESS OTHERWISE SPECIFIED



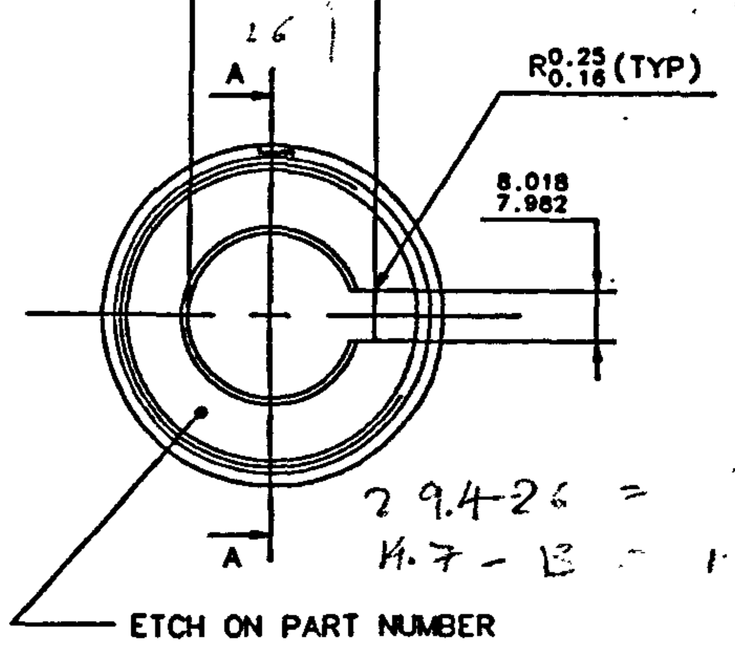
VIEW OF BORE UNDERCUT
SCALE 5:1

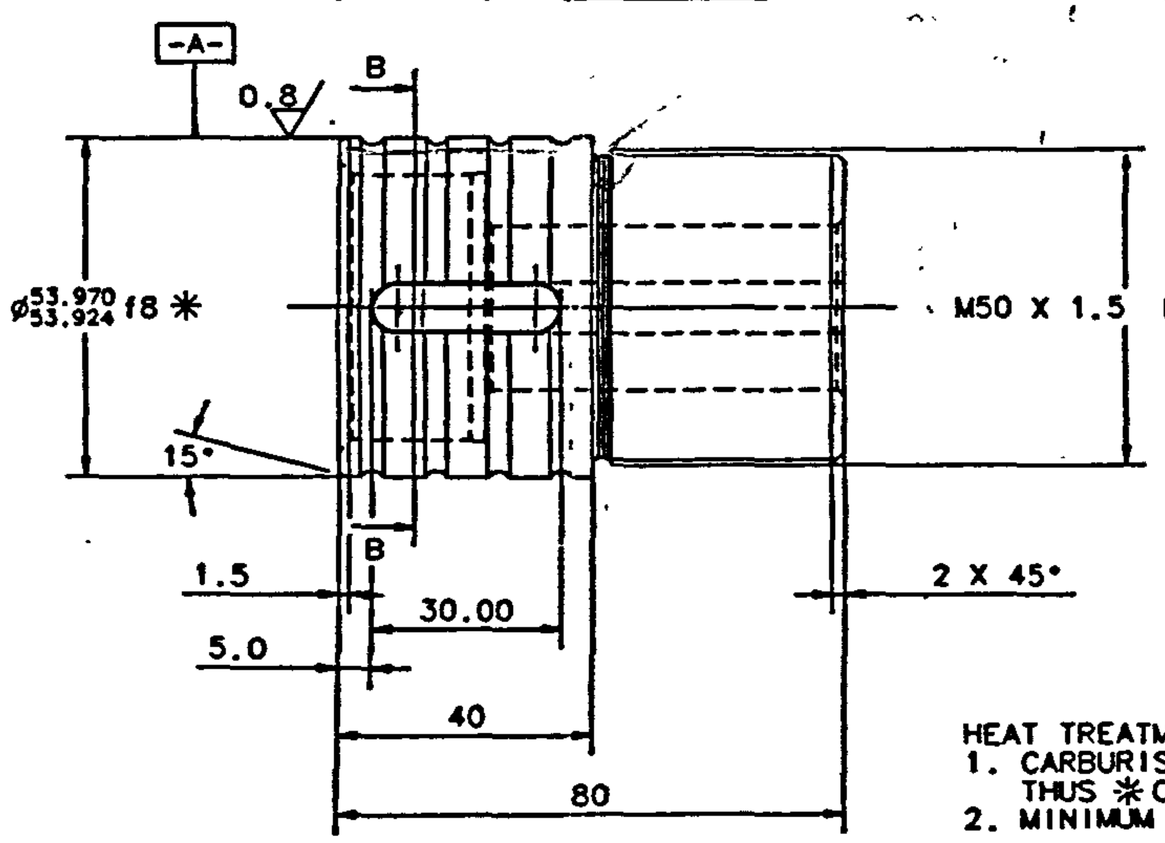


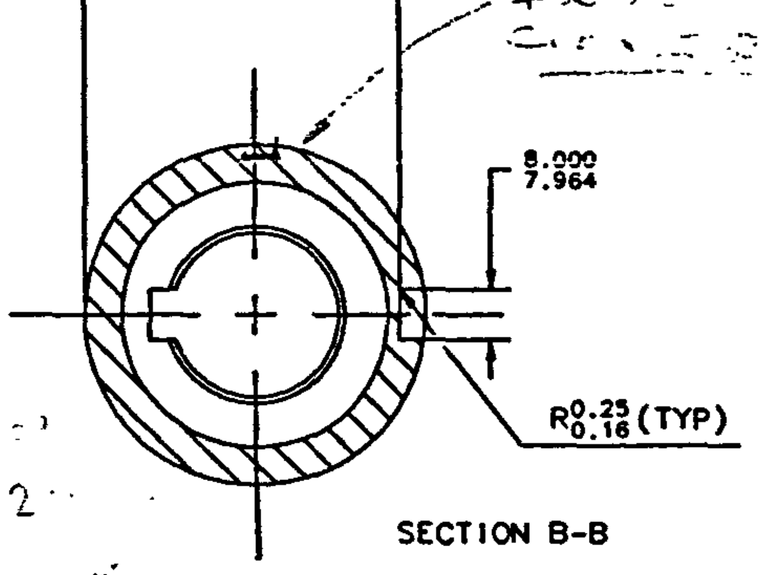
VIEW OF THREAD
UNDERCUT, SCALE 5:1



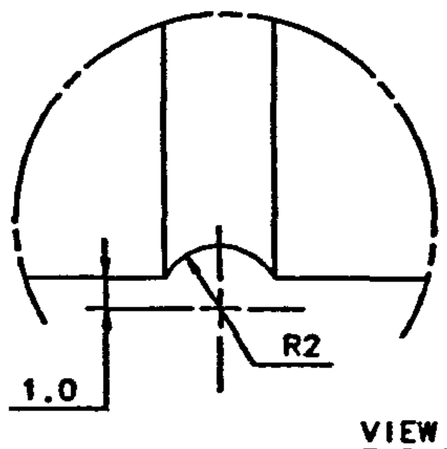
SECTION A-A







SECTION B-B



VIEW OF GREASE GROOVE
TYPICAL AT 4-PLACES
SCALE 5:1

HEAT TREATMENT NOTES:
1. CARBURISE DIAMETERS MARKED THUS * ONLY. HARDEN & TEMPER.
2. MINIMUM HARDNESS Rc58

ALL OVER UNLESS SHOWN OTHERWISE

DESIGN ISSUE 51

Alt No.	Date	Grid	Alteration	Mod. Note

Tolerances
Dimensions to one decimal place (0.0) ±0.25mm
Dimensions to two decimal places (0.00) ±0.1mm
Dimensions to three decimal places (0.000) ±0.025mm
All other dimensions ±0.5mm unless stated

Surface Finish
Machining shown thus ✓ implies a surface finish of 3.2 µm CLA.
Where applicable other CLA values are shown above the symbol in microns.

Drawn MJC
Date 30/10/95
Checked MJC
Machine ORBIT

Material STEEL EN34

Canmaking Systems Shipley England

Issue
Microfilmed

Title: SPRING PLUNGER

Plot Scale 1:1 | Sheet 1 of 1 Sheets

Patts.No. N/A

2001381 C

2001099 10

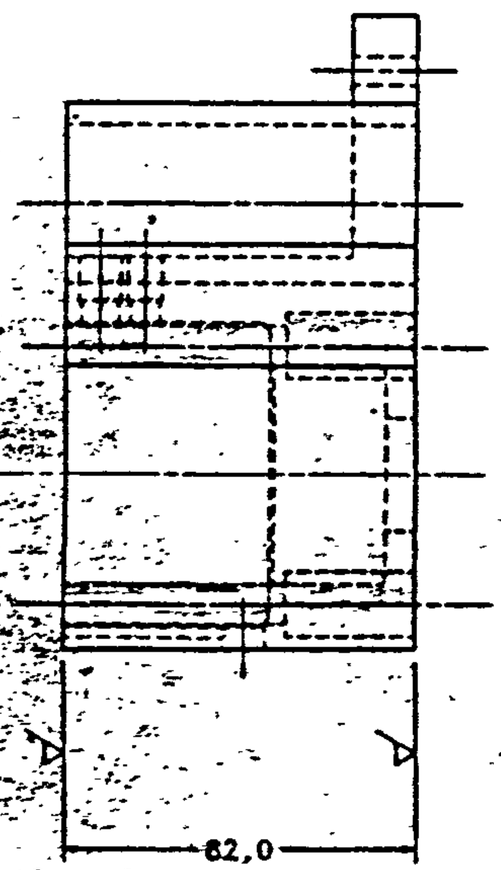
Sheet 1 of 1 Sheets

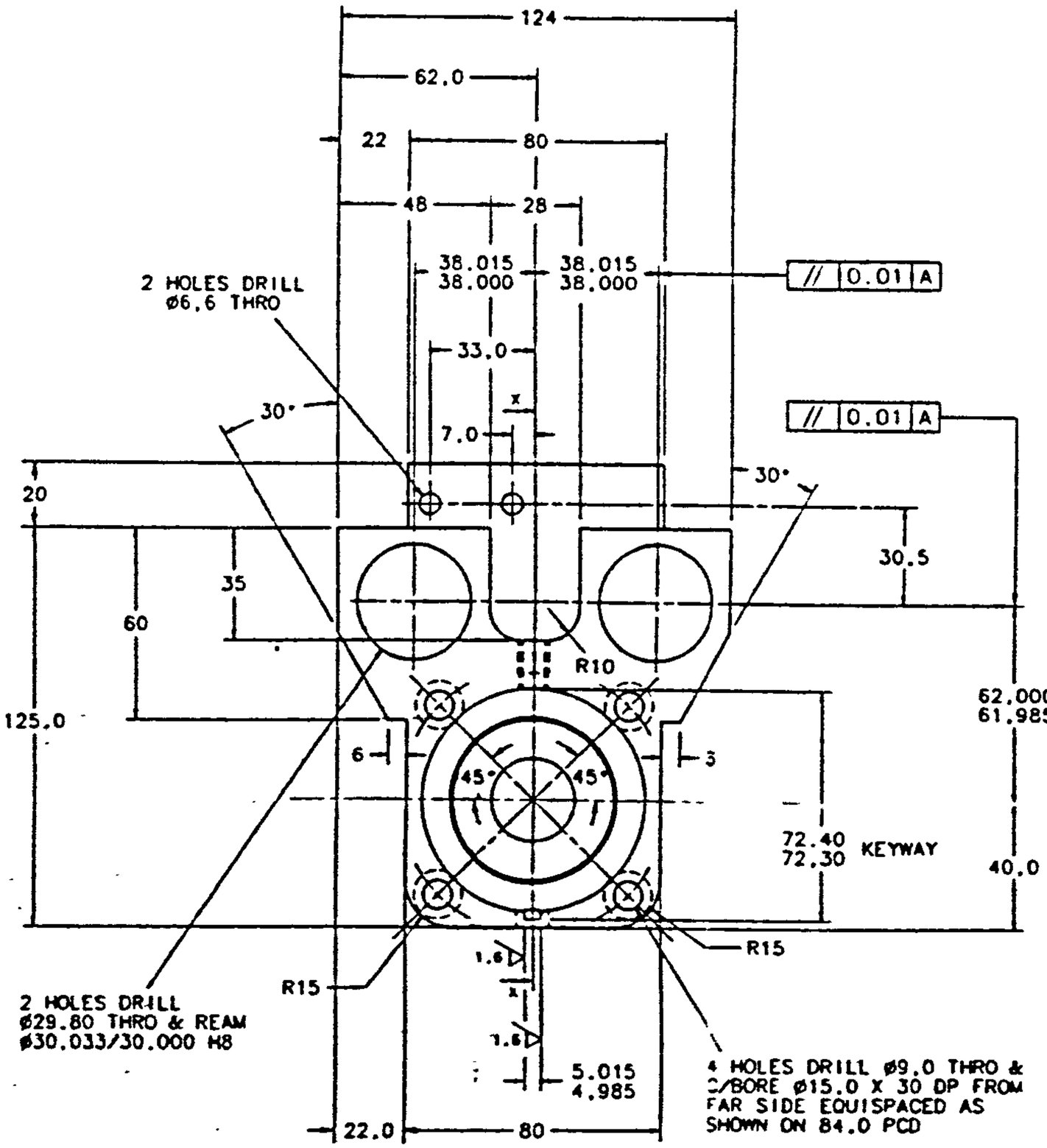
PROJECTION

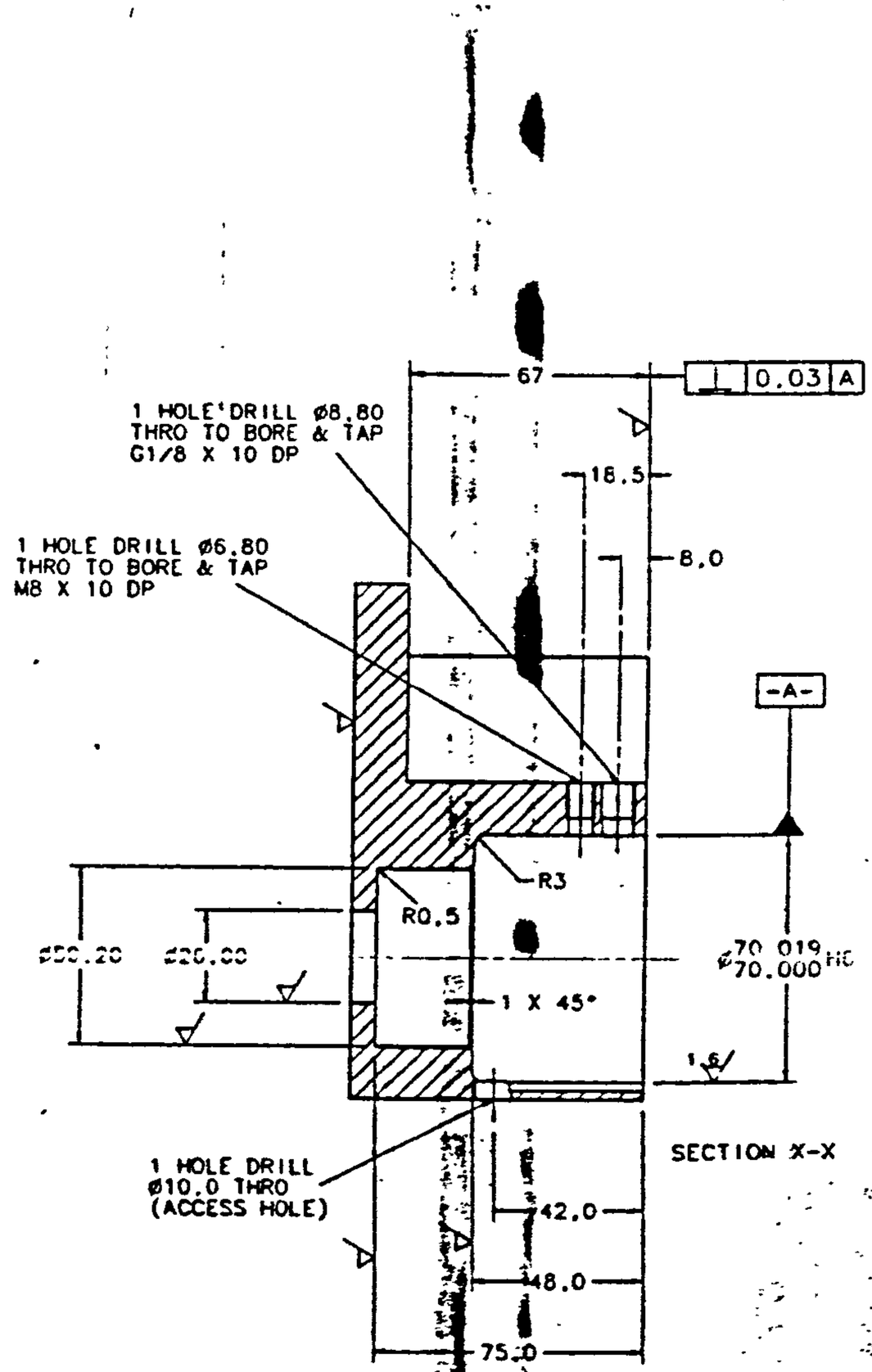
DO NOT SCALE

IF IN DOUBT-ASK

DIMENSIONS IN MILLIMETRES UNLESS OTHERWISE SPECIFIED







SECTION X-X

NOTE: (1) CASTINGS TO BE SUPPLIED TO H.I.P. OF CHESTERFIELD FOR REMOVAL OF THE POROSITY BY THE DENSAL PROCESS.

(2) CASTINGS THEN TO BE HEAT TREATED TO THE TF CONDITION. (SEE PROCEDURE NOTE)
HARDNESS TO BE TESTED AT 10Kg V CKERS
MINIMUM HARDNESS REQUIRED - 110HV
CERTIFICATE IS REQUIRED TO CONF RM THAT THESE CASTINGS WERE TESTED AND ACHIEVED THE SPECIFIED HARDNESS.

(3) FOUNDRY TO PROVIDE BOTH CERTIFICATES ON DELIVERY OF EACH BATCH OF CASTINGS.

(4) BEAD BLAST (NO FERROUS OXIDES TO BE USED)

PROCEDURE FOR HEAT TREATING LM25 TO THE TF CONDITION

SOLUTION TREATMENT - 4 TO 12 HOURS AT 525°-545°C
HOT WATER QUENCH
PRECIPITATION TREATMENT - 8 TO 12 HOURS AT 155°-175°C

CAST PART No ON FACE TO BE MACHINED
ALL UNDIMENSIONED FILLET RADII TO BE R4 INTERNAL AND R10 EXTERNAL.

DESIGN OFFICE COPY

NOTE:
MATERIAL FOR TEST RIG ONLY
TO BE ALUMINIUM ALLOY 6082T6.

4 DEC 1995

DESIGN ISSUE 55

Concentric Hydraulics Engineering plc

SEAMING FOIL HSG

ALUMINIUM BS-1490-LM25-TF

2001099 10

1991003 F

IF IN DOUBT-ASK

ENLARGED VIEW OF OIL GROOVES
SCALE 2:1

PART SECTION "R-R"

PART SECTION "T-T"
SCALE 2:1
TYPICAL OF 16 PLACES

2 HOLES DRILL
ø4 20 X 20 DP
& TAP M8 X 13 DP

ø1260
FLAMECUT

ø1210.00

592.0
4 PLACES

615.0

50.0

27.0

15°
4 PLACES

FARSIDE
FACE

0.5 X 45°
CHAMFER
2 PLACES

ø48 FLAMECUT
16 PLACES

291.014

ø230

ø260

CC

AA

388.899

194.950

185.731

ø270
FLAMECUT

582.029

290.032
290.000

STATION
Ano

MARK IS HIGH NUMBERS
ADJACENT TO EACH ø80 H6
BORE 1-16 AS INDICATED

PART VIEW ON ARROW "A"
SCALE 1:1

MARK 5 HIGH NUMBERS TP1-TP16
ADJACENT TO ø20 H8 BORES AS SHOWN.
DISPLACED AS SHOWN TO STATION Ano.

PART VIEW ON ARROW "A"

PART SECTION "B-B"
SCALE 1:1

PART SECTION "J-J"
SCALE 1:1
TYPICAL OF 16 PLACES

ENLARGED VIEW FROM FARSIDE
SCALE 1:1
(TYPICAL OF 16 PLACES)

81.753
+76.025
+76.000

ø365

ø290

ø88.000

ø87.000

ø29.756

SECTION "H-H"

NOTE:
HOLES & GROOVE MARKED ⊕ TO BE EQUISPACED
WITH A NON-ACCUMULATIVE PITCH OF 185.731
DATUM FOR ALL DOWEL HOLES & BORES FROM
HUB BORE (1991113-03) & NOT TURRET BORE

NOTE ①
THIS HOLE IS OFFSET
& ON A DIFFERENT PCD

A - 3 HOLES DRILL ø6.00 X 25 DP & TAP M8 X 17 DP
EQUISPACED AT FARSIDE AS SHOWN ON 93.0 PCD
- AT 16 PLACES -

B - 3 HOLES DRILL ø6.00 X 25 DP & TAP M8 X 17 DP
EQUISPACED AT FARSIDE AS SHOWN ON 85.0 PCD
- AT 32 PLACES -

C - 1 HOLE DRILL ø5.00 X 17 DP & TAP M6 X 9 DP
AS SHOWN - AT 16 PLACES -

D - 1 HOLE DRILL ø8.0 X 54 DP DRILL
ø4.0 THRU TO ø10 H6 BORE AS SHOWN
- AT 16 PLACES -

E - 1 HOLE DRILL ø8.00 X 18 DP & TAP 9/16 X 8 DP
DRILL ø5.0 THRU TO ø80 H6 BORE AS SHOWN
- AT 16 PLACES -

F - 2 HOLES DRILL ø10.0 X 35 DP BORE & REAM
ø10.027/10.000 H8 X 25 DP AT FARSIDE
POSITIONED AS SHOWN

G - 8 HOLES DRILL ø14.00 X 35 DP & TAP M16 X 24 DP
① 7 HOLES SPACED AS SHOWN ON 874.0 PCD & 1 HOLE OFFSET
AS SHOWN ON ø42.0 PCD
- WHEN 1 HOLE OFFSET & ON DIFFERENT PCD

H - 8 HOLES DRILL ø40 DP X 35 DP & TAP M16 X 20 DP
AT FARSIDE EQUISPACED AS SHOWN ON 700.0 PCD

J - 16 HOLES DRILL ø10.0 THRU, C'BORE ø20.50 X 20 DP,
C'BORE ø20.0 X 6 DP, BORE & REAM ø10.033/10.000 H8
THRU, TAP M12 X 1.5 X 18 DP FROM FAR SIDE.
BORE & REAM ø5% 033/26.000 H8 X 9 DP
POSITIONED AS SHOWN

K - 1 HOLE DRILL ø5.0 X 17 DP DISPLACED AS SHOWN
- AT 16 PLACES -

L - 6 HOLES DRILL ø17.50 X 50 DP & TAP M20 X 37 DP
EQUISPACED 3 ON 470 PCD FROM UNDERSIDE 3 ON 520 PCD
FROM TOP (FOR LIFTING PURPOSES)

M - 7 HOLES DRILL ø17.50 X 51 DP & TAP M20 X 38 DP
AT FARSIDE SPACED AS SHOWN ON 350.0 PCD
- WHEN 1 HOLE OFFSET -

N - 2 HOLES DRILL ø6.80 X 21 DP TAP M8 X 15 DP
DISPLACED AS SHOWN - AT 4 PLACES -

P - 2 HOLES DRILL ø8.50 X 25 DP TAP M10 X 20 DP
DISPLACED AS SHOWN - AT 4 PLACES -

R - 4 HOLES DRILL ø25.5 THRU
EQUISPACED AS SHOWN ON 500.0 PCD

S - 1 HOLE DRILL ø4.20 X 18 DP & TAP M5 X 12 DP
DISPLACED AS SHOWN - AT 16 PLACES -

AA - 1 HOLE DRILL ø18.50 THRU,
C'BORE 5 X 45° & TAP M20 X 1.5 X 20 DP FAR SIDE
(FOR ELECTRICS)

BB - 1 HOLE DRILL ø6.80 X 25 DP & TAP M8 X 17 DP SPACED AS
SHOWN AT FARSIDE - AT 16 PLACES -

CC - 1 HOLE DRILL ø11.80 THRU & TAP 9/16 X 14 DP EACH SIDE
(FOR OIL TO RH TURRET)

DD - 4 HOLES DRILL ø5.00 X 23 DP & TAP ⅜ X 16 DP AT FARSIDE
SPACED AS SHOWN ON 60.0 PCD - AT 4 PLACES -

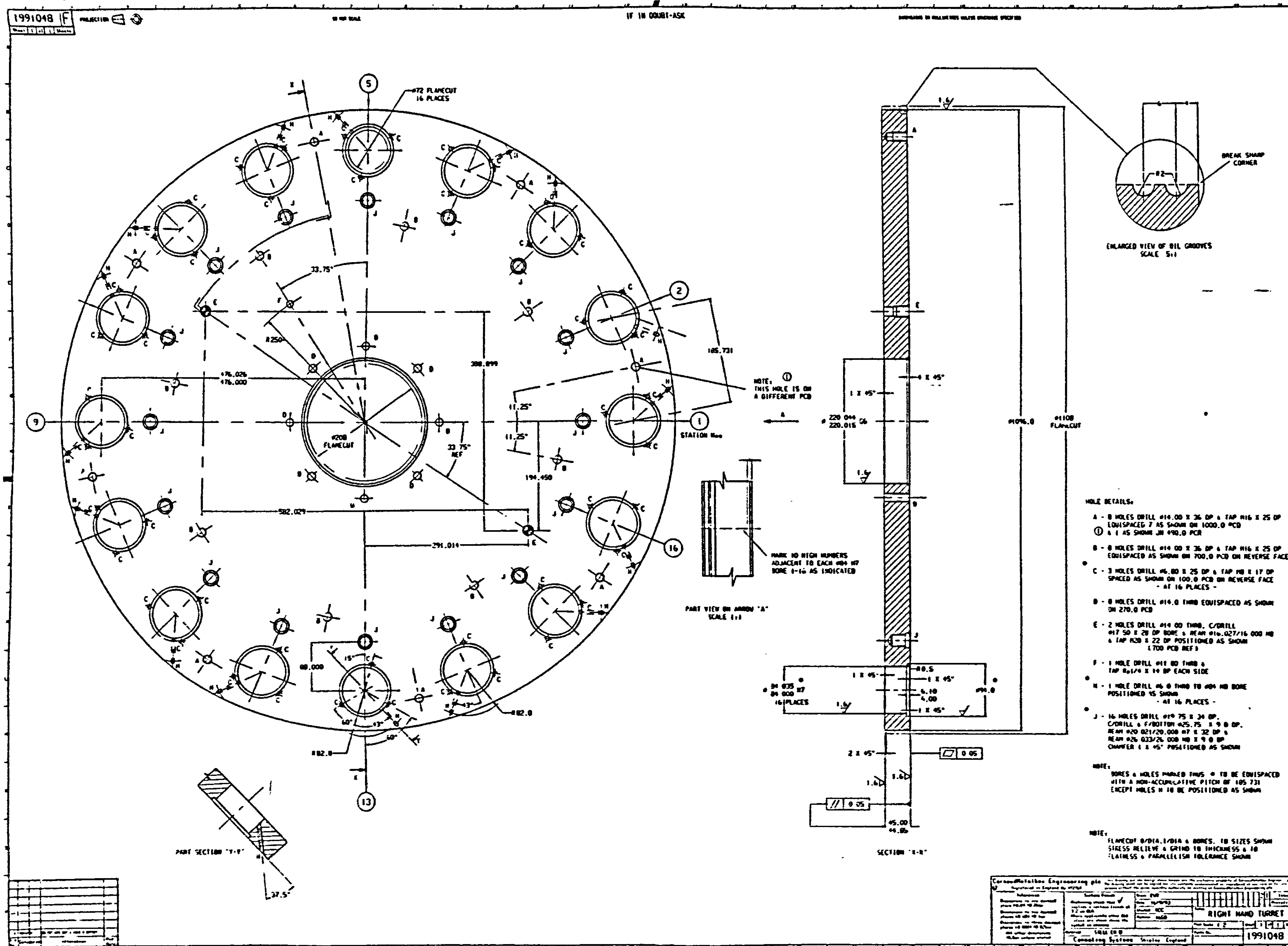
NOTE:
FLAMECUT BORES, FACES & GROOVES
TO SIZES SHOWN.
SURFACE ROUGHNESS & GRADE
TO THICKNESS & FLATNESS &
PARALLELISM TOLERANCE SHOWN

Cammell Laird Engineering plc

LEFT HAND TURRET

Cammell Laird Shipbuilders Limited

1991003 F


1991048 F
IF IN DOUBT-ASK
ø72 FLAMECUT
16 PLACES
ø208
FLAMECUT
NOTE: ①
THIS HOLE IS ON
A DIFFERENT PCD
STATION Nos
MARK 10 HIGH NUMBERS
ADJACENT TO EACH ø84 H7
BORE 1-16 AS INDICATED
PART VIEW ON ARROW "A"
SCALE 1:1
PART SECTION "Y-Y"
SECTION "X-X"
BREAK SHARP
CORNER
ENLARGED VIEW OF OIL GROOVES
SCALE 5:1
HOLE DETAILS:
A - 8 HOLES DRILL ø14.00 X 36 DP & TAP M16 X 25 DP
EQUISPACED 7 AS SHOWN ON 1000.0 PCD
① & 1 AS SHOWN ON 990.0 PCD
B - 8 HOLES DRILL ø14 00 X 36 DP & TAP M16 X 25 DP
EQUISPACED AS SHOWN ON 700.0 PCD ON REVERSE FACE
C - 3 HOLES DRILL ø6.80 X 25 DP & TAP M8 X 17 DP
SPACED AS SHOWN ON 100.0 PCD ON REVERSE FACE
- AT 16 PLACES -
D - 8 HOLES DRILL ø14.0 THRO EQUISPACED AS SHOWN
ON 270.0 PCD
E - 2 HOLES DRILL ø14 00 THRO. C/DRILL
ø17 50 X 28 DP BORE & REAM ø16.027/16 000 H8
& TAP M20 X 22 DP POSITIONED AS SHOWN
(700 PCD REF)
F - 1 HOLE DRILL ø11 80 THRO &
TAP Rp1/4 X 14 DP EACH SIDE
H - 1 HOLE DRILL ø6 0 THRO TO ø84 H8 BORE
POSITIONED AS SHOWN
- AT 16 PLACES -
J - 16 HOLES DRILL ø19 75 X 34 DP.
C/DRILL & F/BOTTOM ø25.75 X 9 0 DP.
REAM ø20 021/20.000 H7 X 32 DP &
REAM ø26 033/26 000 H8 X 9 0 DP
CHAMFER 1 X 45° POSITIONED AS SHOWN
NOTE:
BORES & HOLES MARKED THUS ● TO BE EQUISPACED
WITH A NON-ACCUMULATIVE PITCH OF 185 731
EXCEPT HOLES H TO BE POSITIONED AS SHOWN
NOTE:
FLAMECUT O/DIA.I/DIA & BORES. TO SIZES SHOWN
STRESS RELIEVE & GRIND TO THICKNESS & TO
FLATNESS & PARALLELISM TOLERANCE SHOWN
RIGHT HAND TURRET
1991048 F