CHAPTER – 4
IMPLEMENTATION OF THE MODEL OF THE DESIGN PROCESS
OF METAL CASTINGS

A model of the design process suitable for the effective integration of design process, manufacturability knowledge and expertise has been evolved in the previous chapter. The following criteria must precede the actual implementation of the model of the design process using expert system approach. For simplicity, the term model in this chapter refers to model of the design process of metal castings.

1. Checking the suitability of the model for the expert systems approach.
2. Selection of an appropriate tool.

These considerations are further explained in the subsequent sections.

4.1 COMMON CHARACTERISTICS THE DESIGN MODEL SUITABLE FOR EXPERT SYSTEM APPROACH

The model of the design process of metal castings has been already described in the previous chapter (section 3.3 – 3.5). Some of the characteristic features of the model fact for developing expert systems are as follows:

1. The model is decomposable into small sub-problems which are relatively independent. For example, the component design stage can be scaled down into three sub-problems.
   a) Alloy Selection
   b) Process selection
   c) Selection of pre-engineered functional model for example, the flywheel spoke can be made of rectangular elliptical etc cross-sections. The relative efficiency of these sections or shapes is already known.
   d) Calculation of component dimensions.

The inherent modularity of the model makes it suitable for the expert systems approach.
2. The problem-solving knowledge is precise and well structured for most of the sub-problems. For example, in case of design for manufacturability of metal castings, knowledge is very well structured. Under various headings such as directional solidifications, surface integrity, economical moulding and coring, fettling, machinability etc. Since these sub-problems are well structured, they lend themselves automatically to expert systems development.

3. Some sub-problems such as geometrical modeling and manufacturability analysis are not suitable for the expert systems development. These aspects of the model are cumbersome for knowledge representation because they are computation intensive and need physical skills of the programmer. So, these aspects are best suited for conventional programming techniques. For example, the solidification simulation of Finite Element Method (FEM) is highly computation intensive and is algorithmic in nature. It is better to interface such sub-problems with expert systems.

4.1.2 Suitability of the Entire Model for Expert System Approach

The aspects of the design process for which the expert system development is justified are discussed below:

1. At the beginning, the user defines the requirements of the design. The system involves the user in an interactive dialogue and extracts his requirements of the metal casting. The requirements are translated then into a combination of design variables, constraints and design goals and these need to be checked separately and collectively for consistency. This task is diagnostic in nature. So it can be implemented using standard AI techniques such as decision-lattice structures:

2. Translating the user requirements into a functional model involves selection of an appropriate functional model from a finite set of pre-engineered components which meet the design specifications. This is also diagnostic in nature and involves searching of a repository of component models. Alternatively, the functional decomposition technique can be used to guide the selection of appropriate components. The functional decomposition of a product involves breaking up of the product into smallest distinct sub components and designing each
part one after an other. For example, the design of a flywheel mounted on a shaft involves the design of flywheel mounted on a shaft involves the design of flywheel, shaft and key separately. Further decomposition of flywheel itself can be done by the using of hub, spokes, and rim. Expert systems have widely demonstrated the ability to implement these diagnostic paradigms in domains like casting defect analysis.

3. During the component design stage the component is designed and analyzed to meet the design objectives. The component attributes (design variables) are assigned values and checked for consistency with user requirements. The problem can be designed using top-down refinement of a hierarchical structure.

4. During the component design, the model uses vast amounts of data such as alloy selection. These data can be conveniently separated and stored in data bases. Expert systems can be developed to interface these data bases for easy retrieval and modification. Stand-alone knowledge-based system can also be developed incorporating alloy selection procedures.

5. The geometric modeling extents in many phases of the design model, such as component design, design for manufacturability and manufacturing design during component design the geometrical features are represented from primitives. However, if there is any change in component design, it brings drastic modification into geometrical features. During the design for manufacturability, there will be only minor modifications to the casting geometrical features, eg. Adding fillet radii, taper etc., without affecting component function. During manufacturing design risers, gates, parting line etc., are added to the casting. The geometrical modeling phase also involves physical skills of the designer. Therefore expert system approach is not suitable for geometrical modeling.

6. Design for manufacturability involves assigning quality attributes such as tolerances and surface finish. Simultaneously the geometrical features of the component design are modified to suit the casting process selected. Assigning tolerances and surface finish for the metal casting involves close consideration of factors such as function, alloy process and cost. These problems can be layered into small sub-problems and this inherent modularity makes it suitable for expert systems approach.
As described earlier the manufacturability knowledge is very well structured in the form of abstractions, constraints, heuristics, check-rules and guidelines etc. These knowledge structures are well suited for expert systems development. Symbolic logic is best suited for formalising these structures. These structures based on predicate logic can be implemented in PROLOG. These aspects are explained in detail in Chapter 7.

7. Manufacturing design involves determination of process parameters and design of gates, risers, chills, patterns etc. Though there are several programs available, the foundry men are able to take decisions rapidly and largely without the help of a computer. Heuristic approaches are being used instead of calculations. Since expert systems can readily incorporate heuristics the expert system approach is justified.

8. The geometric models can be interfaced to analysis packages to predict and display the potential errors graphically. Interfacing the expert systems with analysis software has some practical values, such as interpretation of the results. Since the analysis software has some practical values, such as interpretation of the results. Since the analysis packages have already proved successful in day to day foundry operations, it is reasonable to develop interfaces to these packages. However the knowledge based approach to analysis problems is cumbersome, because they are computation intensive.

So the expert systems approach is suitable for many aspects of the model of the design process.

4.2 SELECTION OF AN APPROPRIATE TOOL

After verifying the suitability of the model of the design process, it is important at this stage to select an appropriate tool for implementing the model. However the selection is a difficult task. The various issues that guide the selection of a tool are discussed as under.

The tool spectrum for expert systems development consists of

* Conventional Programming Languages such as Pascal, Fortran, etc.
* Shells like, GURU, VPX, PC +, EXSYS. (Harmon 1985)
* AI languages such as LISP AND PROLOG.
4.2.1 Conventional Programming Languages

Traditional or conventional programming languages such as Fortran, C, Pascal etc can be used for the expert systems development. However, these languages are procedure-oriented and do not support features such as symbolic processing.

These languages require significant amount of time for data management and variable definition. Greater efforts are needed for developing formalisms for knowledge representation and design of inference engine. These tools do not support explanation facilities for example these systems explain how a particular decision is arrived at.

4.2.2 Shells

Shells Knowledge Engineering Languages such as VPX, GURU, OPS 5+ are readily available in the market. The shells are cheaper and much earlier to learn. The expert system development is also faster because they support knowledge base editors, inference engine mechanisms and explanation facilities. The knowledge engineers job is to add the knowledge base to the shell. But there are some inherent disadvantages for using the shells. These are limited to a particular knowledge representation formalisms, inference mechanisms and control strategy. They offer limited flexibility for example if a shell supports rules then it cannot be adopted for frame type of knowledge representation. The systems developed on the shells are also not effective and efficient because of the above mentioned limitations as those developed with a language.

Also available are some powerful and sophisticated knowledge engineering environments, but typically take long time to learn to use. They are more expensive. The other drawback is that there is less programming culture built around each particular environment when compared to general programming languages such as PROLOG. Examples of such environments are LOOPS, KEE.
4.2.3 AI Languages LISP And PROLOG.

AI languages LISP and PROLOG are being used extensively for expert systems development. These languages support flexible symbol manipulation and feature like recursion. In addition, PROLOG supports logical variable, backward chaining and depth first search whereas LISP does not provide the above-mentioned features and to be programmed these features in LISP.

The TURBOPROLOG (TP) compiler has powerful features such as automatic backtracking, pattern matching, tail recursion and multiple internal and external database. Its strength precisely corresponds to the needs of expert systems because TP itself is a language of rules. Even some expert systems like EXDEM, which is implemented in FORTRAN-77 used PROLOG form of rules.

TP offers declarative style of programming and to some extent procedural style also. The extensibility of PROLOG as a programming language makes it suitable for meta-programming, which means it can implement syntax as required by the user. The parsing techniques can be used to interface PROLOG with external packages. The TURBOPROLOG (TP) can also be interfaced to foreign language such as TURBO C, Pascal (User 1986).

TURBOPROLOG (TP) supports most of the requirements of the model of the design process such as:

* The design axioms of component design stages can be expressed in the PROLOG clauses.

* The model makes use of various databases and PROLOG readily supports simultaneous use of multiple databases.

* The expressive power and programming flexibility often make it suitable that any new ideas or changes in design model can easily be implemented during the development stage of the expert system.

* TP also supports graphics which are essential for the design model.

* The programs written in other languages such as TURBO C can be interfaced.
Fig. 4.1: Architecture of Castex System
4.3 IMPLEMENTATION OF CASTEX

The components of CASTEX architecture are shown in Fig. 4.1

1. Inference Engine
2. Knowledge base
   a) rule base       b) working memory
3. (a) Interface to analysis packages
   (b) Analysis packages in C, Pascal
4. (a) Interface to analysis packages
   (b) Analysis package in C, Pascal
5. Explanatory System
6. User Interface.

4.3.1 Inference Engine

TURBOPROLOG is used for developing the inference engine. The inference engine has two components inference and control (Townsend 1988, 1990).

The inference component uses the facts in the working memory and try to trigger new rules. If all the conditions are triggered, then the rule succeeds and is added to the working memory.

The control component determines the order in which the rules are scanned. The search strategies that TURBOPROLOG supports are depth-first search, using backward chaining and limited forward chaining.

In the model of design process the problem-solving is often from general to specific and working top-down hence these search strategies are adequate. In backward chaining, the system assumes a goal and works backward in order to prove it from the facts available in the data base or facts supplied during consultation with the user. If the goal fails the system reverses temporarily, finds another goal and then tries to prove this new goal. This process continues until a goal is proven true or all paths are tested. So, backward chaining systems are also called goal-driven system.
In forward chaining premises of rules are examined against clues from data-base to see if the conclusions are true. If so, the conclusions are added to the database and the cycle repeats. So, systems reasoning forward are called data-driven systems. Since there is firm goal specification in forward chaining it results in combinatorial explosion and the final conclusion is unpredictable.

Forward chaining is appropriate if the number of possible conclusions is very large and input is relatively small. If input information is large and conclusions are a few, backward chaining is appropriate. Although both reasoning strategies can be applied for a given task the choice often depends on time taken to solve a problem. But pure forward chaining is rare in application than backward chaining. Instead a combination backward chaining and limited forward chaining best suited for design domains.

4.3.1 Meta-Rules

Meta-rules are simply rules whose domain of knowledge is the operation of another rule of module. Meta-rules will decide the order of the rules and how to use them. For example, meta-rules are used to select a rule which handles very important constraint. For example, meta-rules to select a rule which handles very important constraint. For example, when wear, corrosion and high temperature are constraints, which are important to apply first and hence will decide the order of the constraints for carrying out search. The advantage of meta-rules is their flexibility and modifiability. They precisely control the search by avoiding combinatorial explosion.

4.3.1.2 Partitioned control structures

When there are more rules in an expert system, the rules can interrelate in many ways, so it is better to partition them into module have minimal interaction with others. An advantage is that the modules can be written and debugged separately.
4.3.2 Knowledge Base

(a) Rule base: Some of the basic structures concerning the design-manufacture, interaction can be represented in the form of conditional IF-THEN Rules which are production rules. These constitute clauses of the system and kept in static memory. For example knowledge structures such as abstractions, heuristics, objects, properties, function, time, causality procedures, algorithms can be represented in the form of PROLOG clauses.

These clauses illustrate the suitability of knowledge structures of design model to TP clauses.

The rules form a hierarchical structure approaching the design process from general to specific in a top-down refinement approach. Heuristics are used to interactively control the reasoning and search process. For example the user may be pursued to answer a multiple choice question and depending on the answer the subsequent reasoning and search are controlled.

4.3.3 Working Memory

working memory stores user requirements, initial design solutions, intermediate and final conclusions which are either true or false. Even data bases can be stored in this dynamic memory: (i) for facts and goals are proved true and (ii) for facts and goals which are proven false. The system queries the user finding out information facts and adding to the knowledge of the working memory. Facts in the database are used to trigger and fire additional rules after which conclusions are added to the working memory. The facts and conclusions stored in the working memory can be interactively changed during consultation. This also causes firing of particular rules.

Since the inference strategy is programmed in TP, any package written in TP can be interfaced easily with a little programming effort. The expert system checks the presence of the package in the disk and prompts the user to load if not available. For a software package, the system should be able to generate the input in order to run it. Similarly the system should be able
to read the output file from the software in order to interpret the results and add information obtained to the knowledge base. To construct a data file in order to operate or read a software file in order to operate or read a software package, the system should be able to extract the relevant information from the knowledge base and format it in an appropriate way. To read the output file from a package, the system needs a parser to understand the structure of the file being read. The parser would then be able to display the results in an appropriate form and make suitable entries to the knowledge base.

4.3.5 Interfacing External Databases

External databases are used as a repository for storing data related both to design and manufacture. The TP supports external databases programming. So data bases written in TP can be easily accessed by the system for fast retrieval of the data. The databases can be accessed directly or via B+ trees. Turbo-Prolog's powerful string handling predicates can be used to create an effective interface for handling databases which are written in text files. These text files can be lexically analysed and modified into a form amenable to the expert system. The interface will transform these text files into a form amenable to the system. During consultation, the system prompts the user to specify the file containing data and attributes of the respective values. The file is scanned and converted to data base form. This data base can be interfaced to the system via inference engine with the help of external database predicates. This data base can be used by the designer for scanning or modifying the values of the attributes or asserting or deleting entries.

Explanatory System

The designer often wants to know why a particular conclusion is arrived and forces the system to display the reasoning chain along with the reasoning process. The trace directive can be used to display the process of satisfying a goal. If the goal is not satisfied, the trace directive can be used to check the failure of a goal and correct the relevant clauses. If new rules and facts are asserted to the knowledge base by knowledge acquisition system (KAS), the explanatory interface to check and maintain the consistency of the knowledge base.
Knowledge Acquisition System (KAS)

The Knowledge Acquisition System is used to assert new rules or facts or edit existing rules. This system checks new, asserted rules for resolving conflicts with the existing rules.

<table>
<thead>
<tr>
<th>Knowledge Structures</th>
<th>Example</th>
<th>PROLOG Clause</th>
</tr>
</thead>
<tbody>
<tr>
<td>Objects</td>
<td>Flywheel</td>
<td>assembly (flywheel, [rim, spokes, hub])</td>
</tr>
<tr>
<td></td>
<td>gear</td>
<td></td>
</tr>
<tr>
<td></td>
<td>air cylinder</td>
<td></td>
</tr>
<tr>
<td></td>
<td>connecting rod</td>
<td></td>
</tr>
<tr>
<td>Properties</td>
<td>Stainless Steel alloy</td>
<td>(stainless, steel, cor.)</td>
</tr>
<tr>
<td></td>
<td>Are corrosion [water, dilute, acids, bases,]</td>
<td>Resistant, sulphur] ; operation (fettling, fettling labor oriented)</td>
</tr>
<tr>
<td></td>
<td>Operation is</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Labour intensive</td>
<td></td>
</tr>
<tr>
<td>Abstraction</td>
<td>Moulding operation</td>
<td>Moulding (x) :- sand mixing,</td>
</tr>
<tr>
<td></td>
<td>Requires ramming, pattern, removal</td>
<td>1. Sand mixing</td>
</tr>
<tr>
<td></td>
<td>2. Ramming</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3. Pattern removal</td>
<td></td>
</tr>
<tr>
<td>Time</td>
<td>Component design</td>
<td>design (x):- component</td>
</tr>
<tr>
<td></td>
<td>Proceeds</td>
<td>design (x), manufacturing</td>
</tr>
<tr>
<td></td>
<td>Manufacturing</td>
<td>design (x)</td>
</tr>
<tr>
<td></td>
<td>Design</td>
<td></td>
</tr>
<tr>
<td>Probability</td>
<td>The probability of</td>
<td>cost (X, tolerance,</td>
</tr>
</tbody>
</table>

50
Attaining tolerances with component (X, tolerance, investment casting is 99.4% tight)

Causality: (1) Precise tolerances cause extra cost.
(2) If the operating temperature of component exceeds by 40% or greater then creep strength is to be considered.

Procedure:
- For surface integrity of a casting:
  1. avoid abrupt changes in cross section
  2. design junctions sections (X)
  3. design joined sections and ribs
  4. design external corners

Quantitative Knowledge Function:
- The thickness of solidified Layer is equal to square root of the time multiplied by a constant X = K Tau.
- Calculate (T,X) :- check (alloy, steel)
  - (Tmin, X = 0.684)
  - sqrt(X)

Taper = - 0.0164 W + 0.0648

W = width, t = thickness