CHAPTER-3

Software Development Tools

Digital design needs optimization or optimality. As discussed in chapter 1 earlier, the Components available for alternative solutions are so enormous that it is humanly very difficult (if not impossible) to scan through the available data. In addition, different technologies and products require different design approaches that again potentially consume a lot of design time and resources if tried manually. To evaluate and select the better alternative is itself a complex job and if attempted manually, will invariably lead to sub-optimal solutions.

The design process will be more efficient with computer aid readily available. It has been advocated in the introduction that the designer stands to gain a lot from the computer help. The computer with reasonable capabilities and resources is available at a low price that is affordable by any office (even by individuals). This adds to the convenience of the design engineer.

The computer system can be helpful and friendly only if good software packages are developed, that aid the designer generating various alternatives based on the available components and help selecting the optimal solution under the constraints of the design. This focuses on the need of a suitable software, using which such packages can be developed.
The software development can be based on the algorithmic approach or the non-algorithmic approach. Each approach is suitable under different conditions. So both approaches have been used.

The database helps in making available, the characteristics of various components. The procedural languages that are useful for program development can be PASCAL, C, FORTRAN, ADA or APL. Among these versatile languages, C has been considered to be best suited for algorithmic program development. Expert system is another suitable area for the system development where non-procedural programming is a necessity. PROLOG is a suitable language for non-procedural programming. It is invariably used for Expert System development.

3.1 C - Language:

C is a general purpose programming language used for the development of system software (e.g. UNIX O.S.) as well as general applications in major non-numerical, text processing and data-base programs. It is known as low-level language replacing the use of assemblers to write efficient and compact programs permitting close interaction with the inner working of computer hardware. At the same time it permits programs to be written the way they are written in other languages i.e. FORTRAN, PASCAL etc. Although C matches the capabilities of many computers, yet it is independent of machine architectures. So the programs written on a computer system can be easily ported on to any other computer system of different architecture.
Again, C does not provide direct operations for composite objects like character strings, sets, lists or arrays and does not provide any input-output facilities, it keeps this language modest and small. The fundamental data objects are characters, integers of several sizes and floating point numbers (with single or double precision). The derived data types of arrays, structures, unions and functions are created with the help of pointers. All other facilities are provided by a standard library of functions. C allows the users to develop their own function libraries. Any function may be called recursively. The functions of a C program may be compiled separately. C does not make any distinction between a function or a subroutine. Variables may be internal to a function, external but known to within a single source file or completely global. Automatic variables may be placed in registers for increased efficiency.

C provides basic flow control constructs; statement grouping; decision making (if-else); looping (do, while & for,) and selection (switch) etc. to provide for structured programming. Pointers provide a strong ability for address arithmetic. Recursion is another excellent feature. C provides for arithmetic, logical and relational operators. Even bitwise operations are available. It has the facility of calling programs written in other languages directly.

C is a developing language and so is available in many versions. It is available even under DOS. However the best performance of C comes under UNIX only. The C++ version is meant for distributed processing.
3.2 Expert Systems:

Artificial Intelligence has been the topic of research for long and has been helped a lot by increased computational power of the present day computers. Problems like natural languages translation, speech understanding, robotics and vision have been worked out quite extensively. Expert systems is one area of Artificial Intelligence that has successful approximate solution to the problem of programming Intelligence. Basically, Expert System is a computer system that emulates the decision making capability of a human expert. These have been successfully applied in business, medicine, science and engineering. Expert systems make extensive use of specialized knowledge (acquired through books, magazines or knowledgeable persons) to solve problems at the level of human expert.

Expert system consists of two parts; Inference Engine and Knowledge Base (specific to a domain). The Inference Engine uses the knowledge from this base to draw conclusions in response to users’ queries, the way a human expert would infer a conclusion. In fact, the expert systems carry a lot of features like, (i) increased availability of expertise (multiplied on many computer systems), (ii) reduced environmental danger (that is hazardous to humans), (iii) permanance (unlike human beings), (iv) multiple expertise (knowledge from many experts put together), (v) increased reliability (no fatigue, stress or tiredness) (vi) explanation of conclusions, (vii) fast response, (viii) complete and steady response at all times and (ix) intelligent data base. In fact, in the process of putting the knowledge to its base, the knowledge has to be explicitly
known that can be examined for correctness, consistency and completeness for adjustments and improvements in quality.

Developing an expert system is gathering experts' knowledge through dialog and coding this knowledge into a knowledge base. The expert system programs are developed that usually have no algorithmic solution. To be able to explain the reasoning is part of an expert system. The expert system can allow itself to learn rules by induction and add knowledge to the base from experience. The adaptive production system learning can be accomplished by (i) adding new rules to the rule base (ii) Deleting old rules from the rule base and (iii) modifying the existing rules in the rule base. Good features include: (a) listing reasons for and against a particular hypothesis, (b) listing all hypotheses that may explain the observed evidence and explain all the consequences of a hypothesis etc.

3.2.1 Elements of Expert System:

In a rule based Expert System, the knowledge base contains the domain knowledge needed to solve problems coded in the form of rules. It consists of:

(i) User Interface (for communication between the user and Expert System).

(ii) Working Memory (a global data base of facts used by rules).

(iii) Inference Engine (makes inferences by deciding which rules are satisfied by facts and executes the rules with some priority).
(iv) Knowledge Acquisition Facility (an automatic mechanism to add the knowledge to its base).

(v) Explanation Facility (to explain reasoning of the system to the user).

The inferencing can generally be done either by forward chaining (where the conclusions are drawn according to the facts) or by backward chaining (where a hypothesis is reasoned and a potential conclusion is to be proved to the facts). Whenever a fact enters the working memory, rules whose patterns are satisfied are said to be activated or instantiated, are fired according to their time tag. All the rules activated are fired one by one before taking up the next facts. The production rules that encode the knowledge in a declarative form are in the form:

\[
\text{Situation - action} \\
\text{or IF } \text{< antecedent 1 >} \\
\text{< antecedent 2 >} \\
\text{.} \\
\text{.} \\
\text{< antecedent m >} \\
\text{THEN } \text{< consequent 1 with certainty C1 >} \\
\text{< consequent 2 with certainty C2 >} \\
\text{.} \\
\text{.} \\
\text{< consequent m with certainty Cm >}
\]
The IF part of the production rule is called the condition part or left hand side or patterns and the THEN part of the production rules is called the action part or right hand side or actions. The rule base is also called knowledge base and the data base is called global data base working memory, short term memory or context list. The cycling of inference engine is called the select-execute loop, recognize-act loop or situation-action loop. When more than one rules are activated, conflict has to be resolved. Various approaches for conflict resolution can be, rule order, data order, generality order or recency order.

3.2.2 Languages for Expert Systems:

There are specialized languages for expert systems such as LISP, PROLOG and ADA. Although expert systems can be written in C or other procedural languages but inefficiently. Basic difference between languages suitable for expert system and other procedural languages is the focus on representation. The expert system languages allow two levels of abstraction, i.e., data abstraction and knowledge abstraction, separating data from methods of manipulating it, i.e., rules and facts. In case of procedural languages, because of tight interweaving of data and knowledge, programmer has to define the sequence of execution carefully. The execution sequence control is considerably less in case of expert systems. Hence, the separation of knowledge and data allows parallelism and modularity.

Expert systems often rely on pattern matching to guide the execution of programs. While other languages, tools and shells have been practised for expert systems, PROLOG stands as real
expert system language. Expert system can be based on declarative programming, nondeclarative programming, object oriented programming, logic programming and induction based programming. PROLOG (and other version Turbo PROLOG) and efficient logic programming tools allowing forward chaining and backward chaining to achieve subgoals for final goal of the problem.

3.3 PROLOG:

It is an accepted language for logic programming and so also for developing the expert systems. It is a non-procedural language and is adaptable to parallel computations. Prolog has very simple acceptable programming rules. Computer programming in Prolog consists of:

* Declaring some facts about objects and their relationships.
* Defining some rules about objects and their relationships.
* Asking questions about objects and their relationships.

Prolog is a conversational language where the user and computer carry out a sort of conversation. Some Prolog fundamentals can be as follows:

Facts are statements represented in a specific way. A fact consists of objects and a relationship i.e. relationship (object;....object), where names of relationships and objects must begin with a lower case letter. The relationship is written first, followed by objects (in parenthesis) separated by commas. The full stop character, "." must come at the end of a fact. The order of objects in parentheses is very important. The name of the relation-
ship that comes before parentheses is called Predicate and each of the objects within parentheses is called Argument.

If we add special symbols before a fact, it becomes a question. The special symbol can be a question mark followed by a hyphen. Prolog searches its database for the question to match the facts. If found in the same order for some predicates, it answers yes; otherwise no if no fact is matched with this in the database.

In Prolog, a variable starts with a capital letter. Initially, the variable is not instantiated. When it stands for an object, it is said to be instantiated. In case of search in database, if a variable is instantiated, user can terminate the search by typing in RETURN or can continue the search to find the alternative instantiations by typing a semicolon.

Conjunctions allow combining many goals together. The AND operation is carried out by a comma and the OR operation by a semicolon. In case of ANDED goals, Prolog searches for the first goal. If the first goal is found in the database, it marks that place in the database and attempts to satisfy the second/subsequent goal. Again if found, Prolog marks its place in the database separately and so on. If however, the second/subsequent goal is not satisfied, then Prolog attempts to resatisfy the previous goal starting from a place marked by the place marker for this sub-goal, thus showing the backtracking capability.

In Prolog, rules are used to say that a fact depends on a group of other facts. A rule is a general statement about objects and their relationships. A rule consists of head and body connected
by :- (a colon followed by a hyphen) and is pronounced "if", left hand side being head and right hand side being the body. To satisfy a rule, Prolog searches the data base to match the head of the rule and then the facts in the body.

Prolog supports arithmetic operators and relational operators. It supports recursion, a very powerful technique for non-numerical programming. It supports Trees and List structures. Cut is another strong facility that allows the user to tell Prolog which previous choices are not to be considered again while backtracking. This allows the programs to work faster and occupy less memory.

Prolog has elaborate input output facility through read and write predicates with punctuation. The reading and writing of terms, characters or sentences is possible (through read, write, put, get and print etc.). Reading and writing the files is done in the similar way except that the current input/output stream has to be changed to the file instead of terminal. The "tell" predicate changes the output stream to a file so that write, put etc. write into the file. The reading from a file is by changing input stream from a file using "see" predicate. The "consult" predicate can read clauses of a program into data base.

Different versions of Prolog have enhancements done to the standard prolog compilers. They add to the capabilities of the language but reduce the portability of programs. Turbo prolog has been used here for software development base.