1. Introduction to Expert Systems

Expert systems are computer programs that are constructed to do the kinds of activities that human experts can do such as design, compose, plan, diagnose, interpret, summarize, audit, give advice. The work such a system is concerned with is typically a task from the worlds of business or engineering/science or government.

Expert systems are usually set up to operate in a manner that will be perceived as intelligent: that is, as if there were a human expert on the other side of the video terminal. A characteristic body of programming techniques give these programs their power. Expert systems generally use automated reasoning and the so-called weak methods, such as search or heuristics, to do their work. These techniques are quite distinct from the well-articulated algorithms and crisp mathematical procedures more traditional programming.

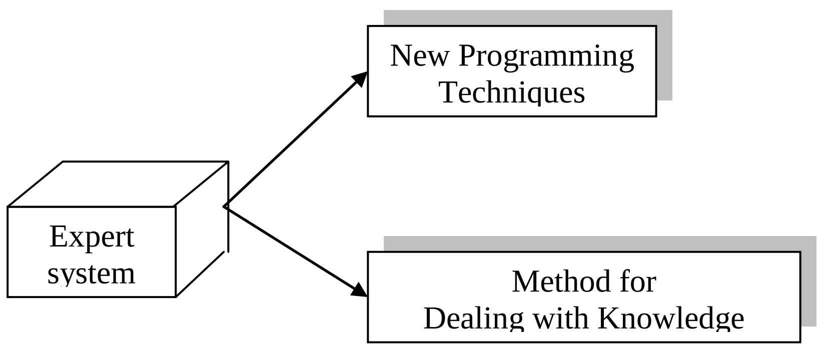


Figure (1) the vectors of expert system development

As shown in Figure(1), the development of expert systems is based on two distinct, yet complementary, vectors:

a. New programming technologies that allow us to deal with knowledge and inference with ease.

b. New design and development methodologies that allow us to effectively use these technologies to deal with complex problems.

The successful development of expert systems relies on a well-balanced approach to these two vectors.

2. Expert System Using

Here is a short nonexhaustive list of some of the things expert systems have been used for:

- To approve loan applications, evaluate insurance risks, and evaluate investment opportunities for the financial community.
- To help chemists find the proper sequence of reactions to create new molecules.
- To configure the hardware and software in a computer to match the unique arrangements specified by individual customers.
- To diagnose and locate faults in a telephone network from tests and trouble reports.
- To help geologists interpret the data from instrumentation at the drill tip during oil well drilling.
- To help physicians diagnose and treat related groups of diseases, such as infections of the blood or the different kinds of cancers.
- To help navies interpret hydrophone data from arrays of microphones on the ocean floor that are used t\u the surveillance of ships in the vicinity.
- To examine and summarize volumes of rapidly changing data that are generated too last for human scrutiny, such as telemetry data from landsat satellites.

Most of these applications could have been done in more traditional ways as well as through an expert system, but in all these cases there were advantages to casting them in the expert system mold.

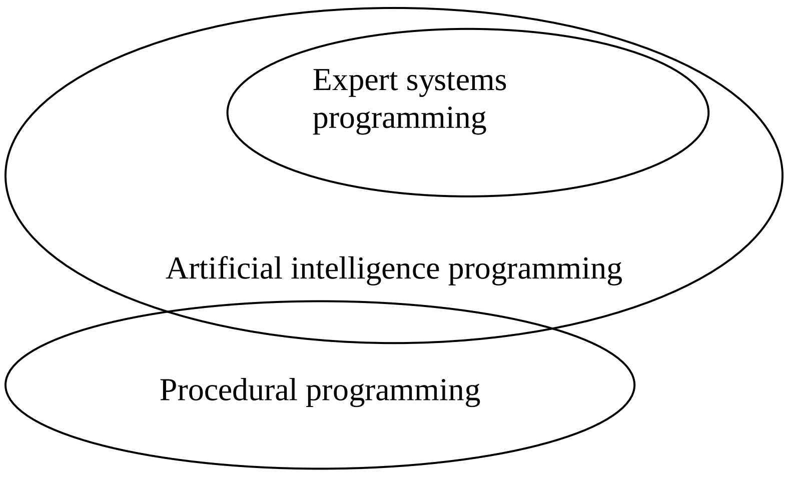
In some cases, this strategy made the program more human oriented. In others, it allowed the program to make better judgments.

In others, using an expert system made the program easier to maintain and upgrade.

3. Expert Systems are Kind of AI Programs

Expert systems occupy a narrow but very important corner of the entire programming establishment. As part of saying what they are, we need to describe their place within the surrounding framework of established programming systems.

Figure(2) shows the three programming styles that will most concern us. Expert systems are part of a larger unit we might call AI (artificial intelligence) programming. Procedural programming is what everyone learns when they first begin to use BASIC or PASCAL or FORTRAN. Procedural programming and A.I programming are quite different in what they try to do and how they try to do it.


Expert systems programming
Artificial intelligence programming
Procedural programming

Figure(2) three kinds of programming

In traditional programming (procedural programming), the computer has to be told in great detail exactly what to do and how to do it. This style has been very successful for problems that are well defined. They usually are found in data processing or in engineering or scientific work.

AI programming sometimes seems to have been defined by default, as anything that goes beyond what is easy to do in traditional procedural programs, but there are common elements in most AI programs. What characterizes these kinds of programs is that they deal with complex problems that are often poorly understood, for which there is no crisp algorithmic solution, and that can benefit from some sort of symbolic reasoning.

There are substantial differences in the internal mechanisms of the computer languages used for these two sorts of problems. Procedural programming focuses on the use of the assignment statement (" = " or ":-") for moving data to and from fixed, prearranged, named locations in memory. These named locations are the program variables. It also depends on a characteristic group of control constructs that tell the computer what to do. Control gets done by using

if-then-else	goto
do-while	procedure calls
repeat-until	sequential execution (as default)

AI programs are usually written in languages like Lisp and Prolog. Program variables in these languages have an ephemeral existence on the stack of the underlying computer rather than in fixed memory locations. Data manipulation is done through pattern matching and list building. The list techniques are deceptively simple, but almost any data structure can be built upon this foundation. Many examples of list building will be seen later when we begin to use Prolog. AI programs also use a different set of control constructs. They are :

procedure calls

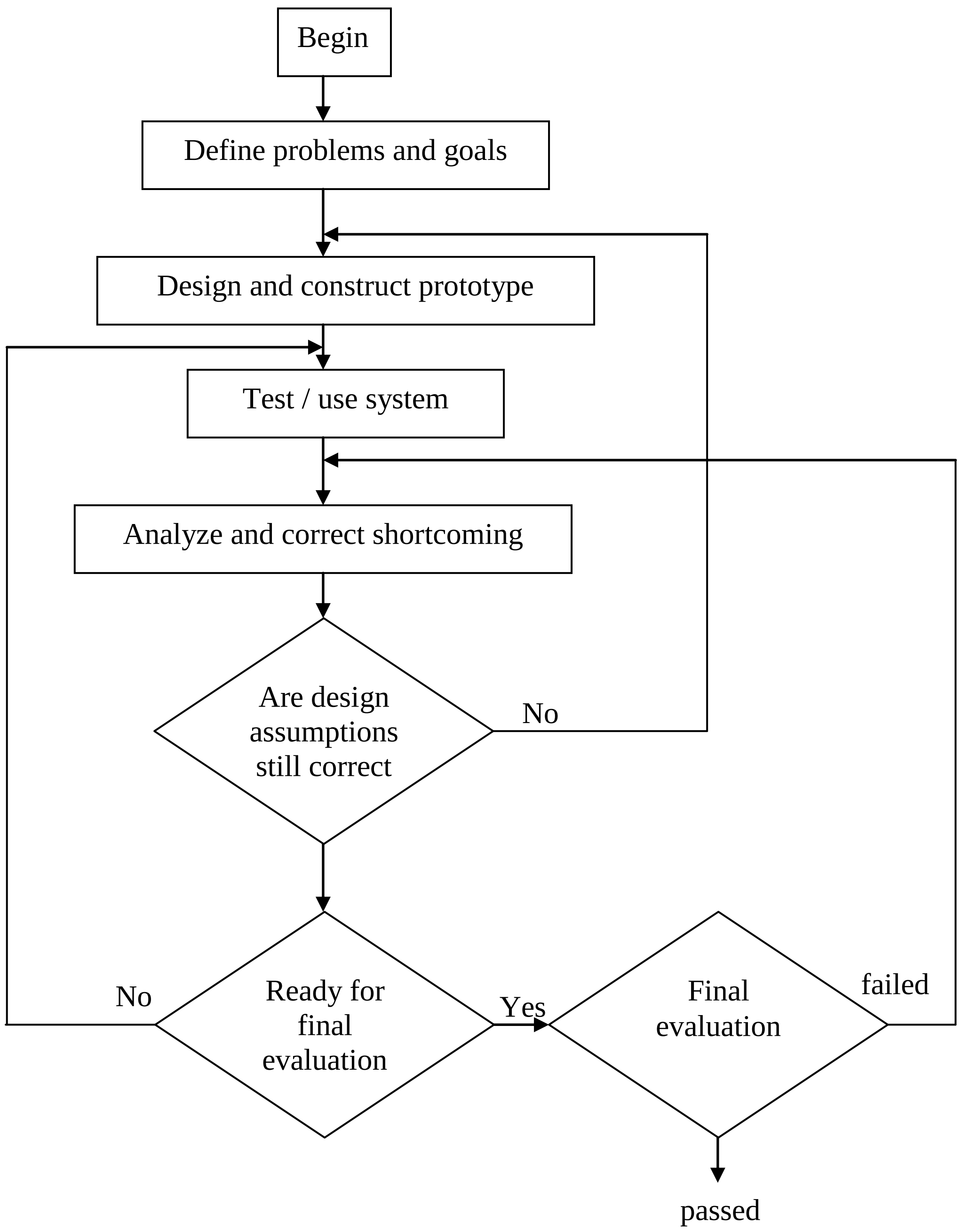
sequential execution

recursion

4. Expert System, Development Cycle

The explanation mechanism allows the program to explain its reasoning to the user, these explanations include justification for the system's conclusions, explanation of why the system needs a particular piece of data.

Figure (3) below shows the exploratory cycle for rule based expert system.

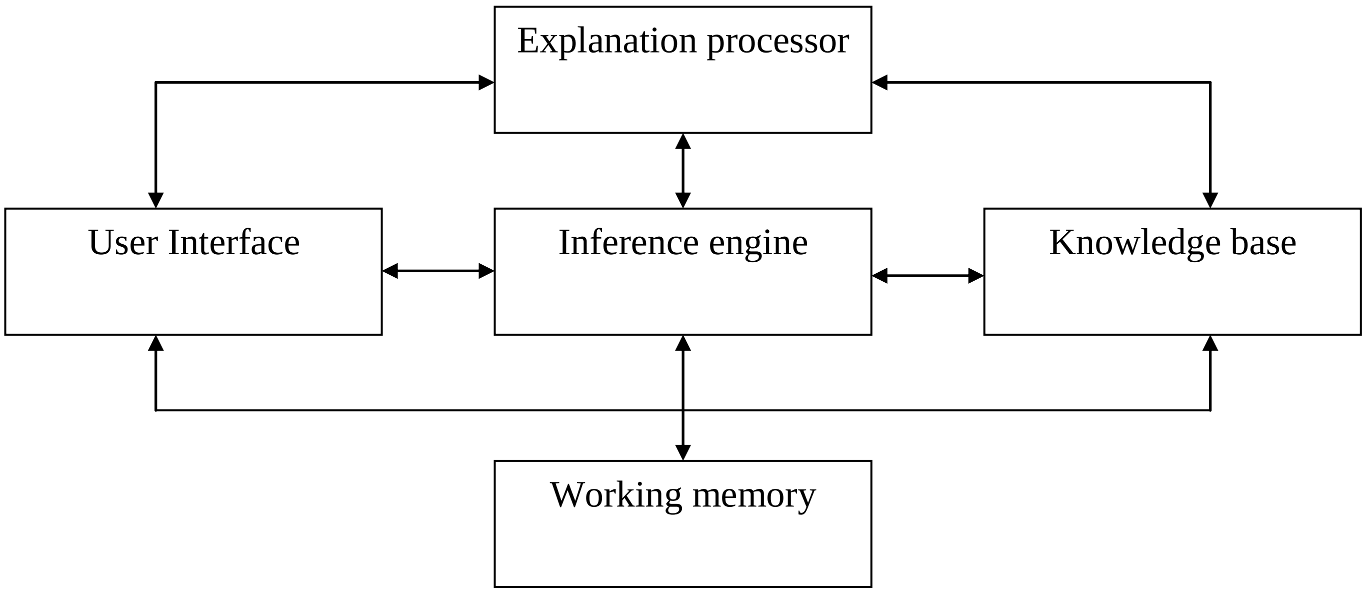


Begin
Define problems and goals
Design and construct prototype
Test / use system
Analyze and correct shortcoming
Are design assumptions still correct
No
No
Ready for final evaluation
Yes
Final evaluation
failed
passed

Figure(3) The exploratory cycle for expert system

5. Expert System Architecture and Components

The architecture of expert system consists of several components as shown in figure (4) below:



Figure(4) Expert system architecture

5.1. User Interface

The user interacts with the expert system through a user interface that make access more comfortable for the human and hides much of the system complexity. The interface styles includes questions and answers, menu-driver, natural languages, or graphics interfaces.

5.2. Explanation Processor

The explanation part allows the program to explain its reasoning to the user. These explanations include justifications for the system's conclusion (HOW queries), explanation of why the system needs a particular piece of data (WHY queries).

5.3. Knowledge Base

The heart of the expert system contains the problem solving knowledge (which defined as an original collection of processed information) of the particular applications, this knowledge is represented in several ways such as if-then rules form.

5.4 Inference Engine

The inference engine applies the knowledge to the solution of actual problems. It s the interpreter for the knowledge base. The inference engine performs the recognize act control cycle.

The inference engine consists of the following components:-

1. Rule interpreter.
2. Scheduler
3. HOW process
4. WHY process
5. knowledge base interface.

5.5. Working Memory

It is a part of memory used for matching rules and calculation. When the work is finished this memory will be raised.

6. Systems that Explain their Actions

An interface system that can explain its behavior on demand will seem much more believable and intelligent to its users. In general, there are two things a user might want to know about what the system is doing. When the system asks for a piece of evidence, the user might want to ask,

"Why do you want it?"

When the system states a conclusion, the user will frequently want to ask,

"How did you arrive at that conclusion?"

This section explores simple mechanisms that accommodate both kinds of questioning. HOW and WHY questions are different in several rather obvious ways that affect how they can be handled in an automatic reasoning program. There are certain natural places where these questions are asked, and they are at opposite ends of the inference tree. It is appropriate to let the user ask a WHY question when the system is working with implications at the bottom of the tree; that is: when it will be necessary to ask the user to supply data.

The system never needs to ask for additional information when it is working in the upper parts of the tree. These nodes represent conclusions that the system has figured out. rather than asked for. so a WHY question is not pertinent.

To be able to make the conclusions at the top of the tree, however, is the purpose for which all the reasoning is being done.   The system is trying to deduce information about these conclusions.   It is appropriate to ask a HOW question when the system reports the results of its reasoning about such nodes.

There is also a difference in timing of the questions. WHY questions will be asked early on and then at unpredictable points all throughout the reasoning. The system asks for information when it discovers that it needs it. The. time for the HOW questions usually comes at the end when all the reasoning is complete and the system is reporting its results.