Expert System

- An **expert system** is a computer system whose performance is guided by specific, expert knowledge in solving problems.
- It is a computer system that simulates the decision- making process of a human expert in a specific domain.
- Expert system is one of the early (large- scale) successes of artificial intelligence.
- An expert system is an “intelligent” program that solves problems in a narrow problem area by using high-quality, specific knowledge rather than an algorithm.
- Expert systems are used by most of the large or medium sized organization as a major tool for improving productivity and quality.
- An expert system’s knowledge is obtained from expert sources and code in a form suitable for the system to use in its process.

Features of an Expert System

- Should have reasoning capacity. Reasoning may be goal-driven reasoning (backward chaining) or data-driven reasoning (forward chaining).
- Should cope with uncertainty.
- Should have proper knowledge representation i.e. use knowledge rather than data.
- Should use symbolic representation for knowledge.
- Should use meta knowledge.
- Should use user interface.
- Should have ability to explain solutions with respect to problem specific.
- Goal driven reasoning (backward chaining) or data driven reasoning (forward chaining)
- Coping with uncertainty
- User interface
- Explanations (ability to explain solution with respect to specific problem)

Advantages of Expert System

- It provides consistent answer for repetitive decisions, processes and tasks.
- Hold and maintained significant level of information.
- Encourage organization to clarify the logic of their decision making.
- Ask question like human expertise.

Disadvantages of Expert System

- Lack of common sense needed in some decision making.
- Cannot make creative response as human expert would in unusual circumstances.
- Error may occur in the knowledge base and lead to wrong decision.
- Cannot adopt changing environment, unless knowledgebase is changed.

Applications

- Business
- Manufacturing
- Medicine
- Engineering
- Applied science
- Military
- Space
- Transportation
- Education
- Image analysis
- Chemical structure
- Agriculture
- Robotics and many more

History of Expert Systems

- Breakthrough in field of AI was achieved in area of Expert System
- 1966 - DENDRAL was developed at Stanford
    - Analyzes mass spectrographic, NMR, … data and infers possible structures of unknown chemicals
- 1971 - MYCIN was developed at Stanford
    - diagnose and treat infectious blood diseases
- 1980 - XCON was developed at CMU
    - one of the first commercial expert systems
- PROSPECTOR: analyzing geological data
- GASOIL: designing gas-oil separation systems for offshore oil platforms

Steps in Expert System Development

Knowledge Acquisition:
- Knowledge acquisition is the process used to define the rules and ontologies required for a knowledge-based system.
- Describe the initial tasks associated with developing an expert system that include finding and interviewing domain experts and capturing their knowledge via rules, objects, and frame-based ontologies.
- The expert sources can be domain specialist, articles, journal, database etc.

Knowledge Representation:
- Representing information about the world in a form that a computer system can utilize to solve complex task.

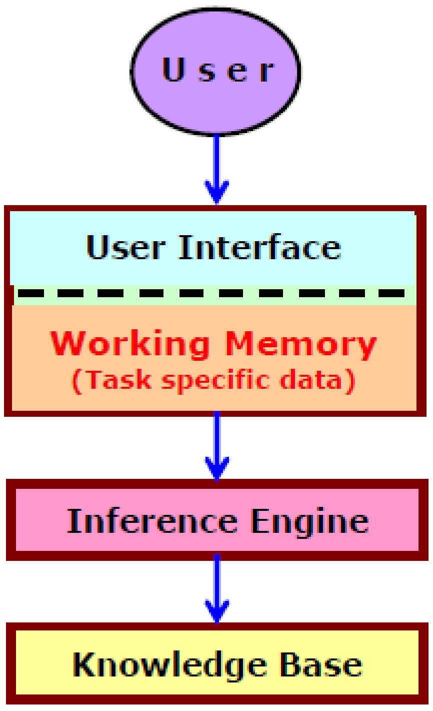
- It is a set of ontological commitments i.e. an answer of question.
- Knowledge representation can be logical representation or structured representation.
- Representing the rules and constraints is through the use of logic rules, formally known as knowledge representation.

Knowledge Inference

- Knowledge inference refers to acquiring new knowledge from existing facts based on certain rules and constraints.
- Mostly rule based reasoning (Forward chaining/Backward chaining) is used for inference

Knowledge Transfer
- Knowledge transfer is the practical problem of transferring knowledge from one part of the organization to another.
- Knowledge transfer seeks to organize, create, capture or distribute knowledge and ensure its availability for future users.

Architecture of a rule based expert system:



User Interface:
The interaction between user seeking a solution to a problem and an expert system takes place through user interface. It takes query from user and displays the result.

Inference engine:
Inference engine applies the knowledge base to deduce new facts and rules. The new facts or rules deduced after application of rule are written to knowledge base. It reads data from case specific data and check which rule can be applied. If more than one rule can be applied, it selects subset of rules based upon some heuristics and execute them. Any change in current state of the problem is written to case specific data. It may use the techniques like backward chaining or forward chaining for reasoning.

Knowledge base:
Knowledge base contains domain specific facts and rules to solve the problem. It is stored in long term memory

Case specific data (working memory):
It is working memory which contains current state of data or current deduction. The user query from user interface directly comes to this component of expert system. It is stored in short term memory.

The main players in the development team:

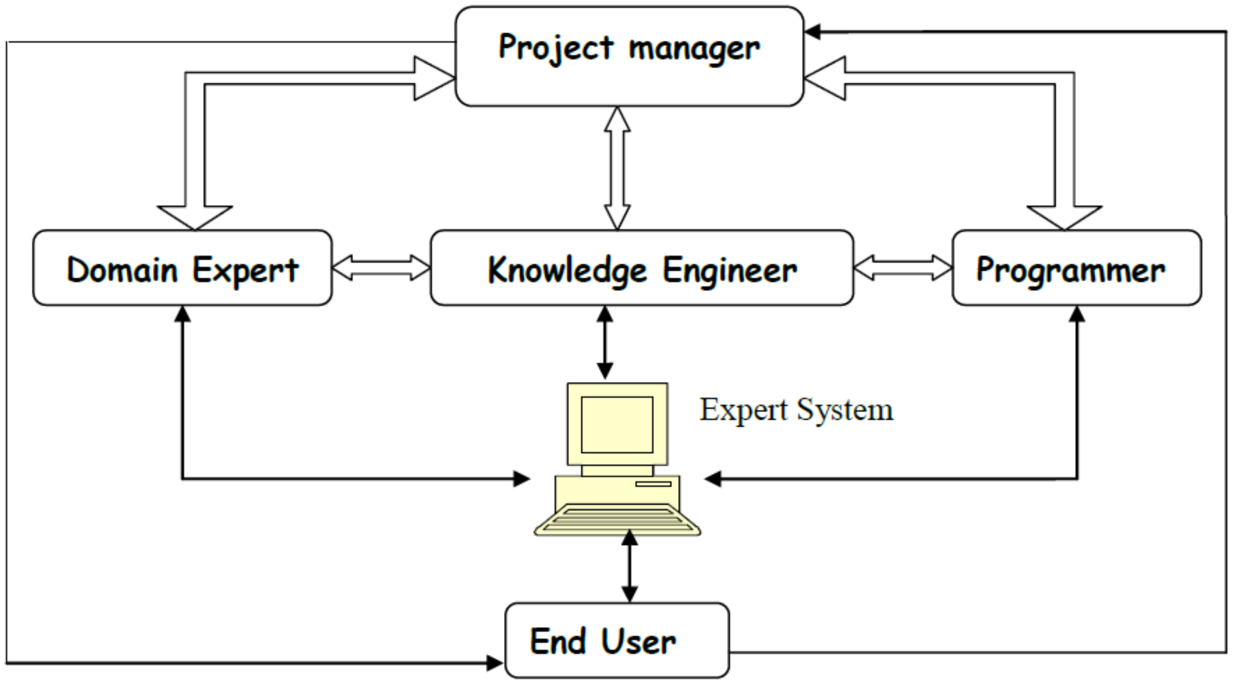
Domain expert:
The domain expert is a knowledgeable and skilled person capable of solving problems in a specific area or **domain. T**his person has the greatest expertise in a given domain. This expertise is to be captured in the expert system.

Knowledge engineer: The Knowledge engineer is someone who is capable of designing, building and testing an expert system. He or she interviews the domain expert to find out how a particular problem is solved.
Knowledge engineer decides how to represent facts and rules in an expert system. The knowledge engineer then chooses some development software. The knowledge engineer is also responsible for testing, revising and integrating the expert system into the workspace. Extracting knowledge from expert, formalizing it as facts and rules, refining it, is called known engineering done by knowledge engineer.

Programmer:
The programmer is the person responsible for the actual programming, describing the domain knowledge in terms that a computer can understand. A programmer must develop the knowledge and data representation structures (knowledge and database), control structure (inference engine) and dialogue structure (user interface) using symbolic languages such as LISP, Prolog and OPS5 or expert system shells.

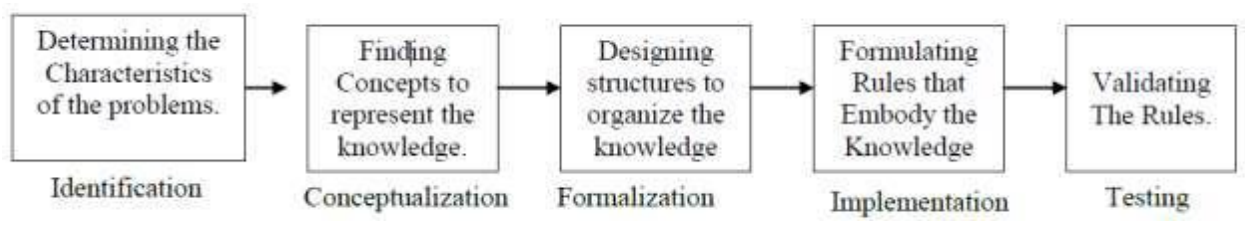
Project Manager:
The project manager is the leader of the expert system development team, responsible for keeping the project on track. He or she makes sure that all deliverables and milestones are met, interacts with the expert, knowledge engineer, programmer and end-user.

End user:
The end-user**,** often called just the user, is a person who uses the expert system when it is developed. Design of the user interface must be in such a way that it satisfies the needs of user and user is comfortable and confident using the system.



Stages of expert system development:

An expert system typically is developed and refined over a period of several years. We can divide the process of expert system development into five distinct stages. In practice, it may not be possible to break down the expert system development cycle precisely. However, an examination of these five stages may serve to provide us with some insight into the ways in which expert systems are developed.


Determining the Characteristics of the problems.
Identification
Finding Concepts to represent the knowledge.
Conceptualization
Designing structures to organize the knowledge
Formalization
Formulating Rules that Embody the Knowledge
Implementation
Validating The Rules.
Testing

Identification:
Beside we can begin to develop an expert system, it is important that we describe, with as much precision as possible, the problem that the system is intended to solve. we must determine the exact nature of the problem and state the precise goals that indicate exactly how we expect the expert system to contribute to the solution.

Conceptualization:
In the conceptualization stage the knowledge engineer frequently creates a diagram of the problem to depict graphically the relationships between the objects and processes in the problem domain. It is often helpful at this stage to divide the problem into a series of sub-problems and to diagram both the relationships among the pieces of each sub-problem and the relationships among the various sub-problems.

Formalization:
In formalization stage, the effort is made to relate the domain problem with AI technology i.e. expert system. After analyzing the identification and conceptualization states, knowledge engineer uses various techniques of knowledge representation and heuristic search used to realize expert systems. The expert system tools like Prolog or CLIPS also can be used that can greatly expedite the development process.

Implementation:
During the implementation stage, the formalized concepts are programmed onto the computer that has been chosen for system development, using the predetermined techniques and tools.

Testing:
Testing provides opportunities to identify the weakness in the structure and implementation of the system and to make the appropriate corrections.

MYCIN

- Expert system for treating blood infections
- Diagnose patients based on reported symptoms and medical test results
- Could ask some more information and lab test results for diagnosis
- Recommend a course of treatment, if requested, MYCIN would explain the reasoning that lead to its diagnosis and recommendation.
- Use about 500 production rules, MYCIN operated roughly the same level of competence as human specialists in blood infections.
- Use backward chaining for reasoning**.**

DENDRAL

- First ES developed in late 1960
- Designed to analyze mass spectra
- Based on the mass of fragments seen in the spectra, it would be possible to make inference as the nature of molecule tested, identifying functional groups or even the entire molecule.
- Use Heuristic knowledge obtained from experienced chemists.
- Use forward chaining for reasoning

EMYCIN

- Emycin is an expert system shell, a framework for building programs that record the knowledge of domain experts and use that knowledge to help non-expert users solve problems.
- It provides an interface that helps experts define data types and rules, a backwards-chaining reasoning algorithm (similar to Prolog, but with key differences), a mechanism for dealing with uncertainty, and facilities for introspection that permit users to learn what the system knows and what it is doing.

Case-based reasoning (CBR)

Case-based reasoning (CBR) is the process of solving new problems based on the solutions of similar past problems. In case-based reasoning (CBR) systems expertise is embodied in a library of past cases, rather than being encoded in classical rules. Each case typically contains a description of the problem, plus a solution and/or the outcome. The knowledge and reasoning process used by an expert to solve the problem is not recorded, but is implicit in the solution.

To solve a current problem: the problem is matched against the cases in the case base, and similar cases are retrieved. The retrieved cases are used to suggest a solution which is reused and tested for success. If necessary, the solution is then revised. Finally the current problem and the final solution are retained as part of a new case.

Case-based reasoning is liked by many people because they feel happier with examples rather than conclusions separated from their context. A case library can also be a powerful corporate resource, allowing everyone in an organization to tap into the corporate case library when handling a new problem.

Since the 1990's CBR has grown into a field of widespread interest, both from an academic and a commercial standpoint. Mature tools and application-focused conferences exist. Case-based reasoning is often used as a generic term to describe techniques including but not limited to case-based reasoning as we describe it here (e.g. analogical reasoning is often referred to as case-based reasoning).

All case-based reasoning methods have in common the following process:

- retrieve the most similar case (or cases) comparing the case to the library of past cases;
- reuse the retrieved case to try to solve the current problem;
- revise and adapt the proposed solution if necessary;
- retain the final solution as part of a new case.

There are a variety of different methods for organizing, retrieving, utilizing and indexing the knowledge retained in past cases.

Retrieving a case starts with a (possibly partial) problem description and ends when a best matching case has been found. The subtasks involve:

- identifying a set of relevant problem descriptors;
- matching the case and returning a set of sufficiently similar cases (given a similarity threshold of some kind); and
- Selecting the best case from the set of cases returned.

Some systems retrieve cases based largely on superficial syntactic similarities among problem descriptors, while advanced systems use semantic similarities.

Reusing the retrieved case solution in the context of the new case focuses on: identifying the differences between the retrieved and the current case; and identifying the part of a retrieved case which can be transferred to the new case. Generally the solution of the retrieved case is transferred to the new case directly as its solution case.

Revising the case solution generated by the reuse process is necessary when the solution proves incorrect. This provides an opportunity to learn from failure.

Retaining the case is the process of incorporating whatever is useful from the new case into the case library. This involves deciding what information to retain and in what form to retain it; how to index the case for future retrieval; and integrating the new case into the case library.

A CBR tool should support the four main processes of CBR: retrieval, reuse, revision and retention. A good tool should support a variety of retrieval mechanisms and allow them to be mixed when necessary. In addition, the tool should be able to handle large case libraries with retrieval time increasing linearly (at worst) with the number of cases.

Applications

Case based reasoning first appeared in commercial tools in the early 1990's and since then has been sued to create numerous applications in a wide range of domains:

- Diagnosis: case-based diagnosis systems try to retrieve past cases whose symptom lists are similar in nature to that of the new case and suggest diagnoses based on the best matching retrieved cases. The majority of installed systems are of this type and there are many medical CBR diagnostic systems.
- Help Desk: case-based diagnostic systems are used in the customer service area dealing with handling problems with a product or service.
- Assessment: case-based systems are used to determine values for variables by comparing it to the known value of something similar. Assessment tasks are quite common in the finance and marketing domains.
- Decision support: in decision making, when faced with a complex problem, people often look for analogous problems for possible solutions. CBR systems have been developed to support in this problem retrieval process (often at the level of document retrieval) to find relevant similar problems. CBR is particularly good at querying structured, modular and non-homogeneous documents.
- Design: Systems to support human designers in architectural and industrial design have been developed. These systems assist the user in only one part of the design process, that of retrieving past cases, and would need to be combined with other forms of reasoning to support the full design process.