Expert Systems

SUKANTA GHOSH

Topics Covered

What is an Expert System?

Why should we use Expert Systems?

History of Expert Systems.

Latest Developments in Expert Systems.

Components of an Expert System.

Rule Based Reasoning in Expert Systems.

Limitations of Expert Systems.

What is an Expert System?

Human experts have

- a considerable knowledge about their areas of expertise
- Can learn from their experience
- Can do reasoning
- Can explain the solution
- Can restructure knowledge
- Can determine relevance

What is an Expert System?

An Expert System (ES) is software that attempts to reproduce the performance of one or more human experts, typically in a specific problem domain.

ES imitate the expert's reasoning processes to solve specific problems.

A computer program designed to model the problem-solving ability of a human expert. (Expert Systems Design and Development by Durkin)

A model and associated procedure that exhibits, within a specific domain, a degree of expertise in problem solving that is comparable to that of a human expert. (Introduction to Expert Systems by Ignizio)

A computer system which emulates the decision-making ability of a human expert. (Expert Systems: Principles and Programming by Giarratano and Riley)

Why should we Use Expert System?

Expert Systems:

- Capture and preserve irreplaceable human expertise
- Provide expertise needed at a number of locations at the same time or in a hostile environment that is dangerous to human health
- Provide unemotional objective solutions faster than human experts
- Provide expertise that is expensive or rare
- Share human expertise with a large number of people
- Aids in training new employees
- Provides second opinion in critical situations.

History of Expert Systems

In 1959 Newell and Simon described General Problem Solver (GPS).

GPS

- Intended to solve general problems across domains. Example: theorem proof, geometric problem and chess playing.
- First computer program to separate knowledge from strategy (generic solver engine).
- Predecessor to ES.

ES introduced by Stanford Heuristic Programming Project led by Feigenbaum.

History of Expert Systems

Mid-1960s: Early ES programs:

MYCIN:

- Diagnose infectious diseases such as bacteremia and meningitis.
- Recommend antibiotics.
- Dosage adjusted for patient's body weight.
- Name derived from antibiotics (suffix – "mycin").

Mid-1970s:

Recognition of the role of knowledge

- Power of an ES comes from the specific knowledge it possesses, not from the inference schemes it employs.
- Development of knowledge representation theories.
- Development of decision making procedures and inferences.

History of Expert Systems

Early 1980s:

Expert systems proliferated. Example: XCON, XSEL.

XCON (eXpert CONfigurator):

- Used to assist in the ordering of DEC's VAX computer system
- Automatically selected the computer system components based on the customer's requirements
- Saved DEC $25M a year by speeding the assembly process and increasing customer satisfaction

XSEL:

- A newer version of XCON
- Intended to be used by DEC's salesforce

History of Expert Systems

Universities offered expert system courses.

ES technology became commercial.

Programming tools and shells appeared. Example: EMYCIN, EXPERT

Criteria for Building an Expert System

Does a human know how to solve the problem?

Does the problem have a definable solution?

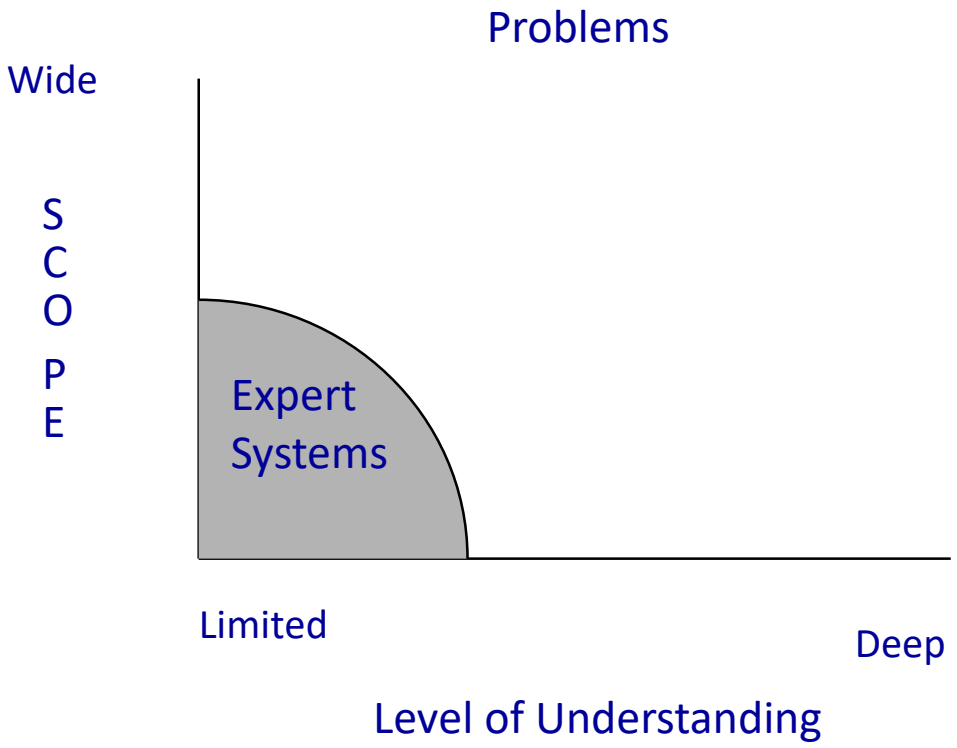
Is the level of understanding and scope appropriate?

Has the technique for solving the problem been documented?

Level of Understanding

Defining the problem to fall within the shaded area of the graph is very important -

Problem should NOT have too large a scope or too deep a level of understanding



Advantages of an Expert System

Rapid prototype development

Easier verification of software

Easier maintenance of software

Explains its reasoning in English to user when requested

Truly self documenting software

Easier to learn to build rule-based expert systems

Inexpensive technology

Automated consistency checking of knowledge in the knowledge base

Disadvantages of an ES

Brittleness

- only have access to highly specific domain knowledge
- cannot fall back on more general knowledge when needed

Lack of Meta-Knowledge

- do not have sophisticated knowledge about their own operation
- cannot reason about their own scope and limitations

Knowledge Acquisition is a bottleneck in applying ES technology to new domains

Validation of an ES is difficult

Expert Systems can make mistakes

Expert Systems vs Conventional Programs

Characteristic	Conventional Program	Expert System
Control by...	Information & control integrated	Knowledge separate from control
Solution by...	Algorithm	Rules & inference
Execution	Generally sequential	Opportunistic rules
Input	Must be complete	Can be Incomplete
Design	Structured	Little or none
Expansion	Done in major jumps	Incremental
Representation	Numeric	Symbolic
Modify	Difficult to modify	Easy to modify
Results	Final result given	Recommendation w/expansion
Interface	Command interface	Natural dialogue
Answers	Optimal Solution	Acceptable solution

Personnel Involved

Domain Expert

- a person who posses some skills that allow him/her to draw upon past experiences and quickly focus on the core of a given problem.

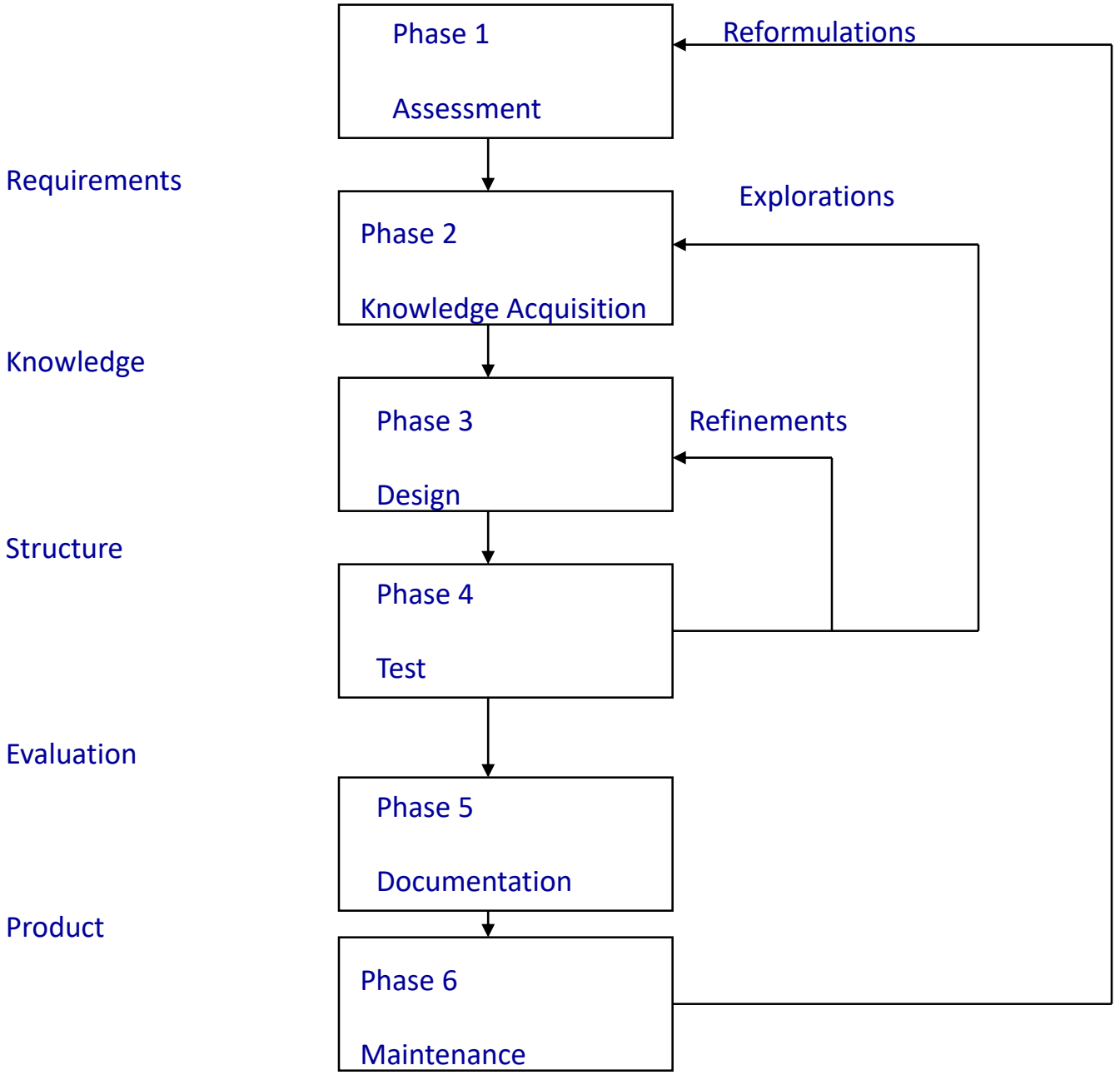
User

- a person who will use the expert system and ultimately benefit from the domain expert's knowledge.

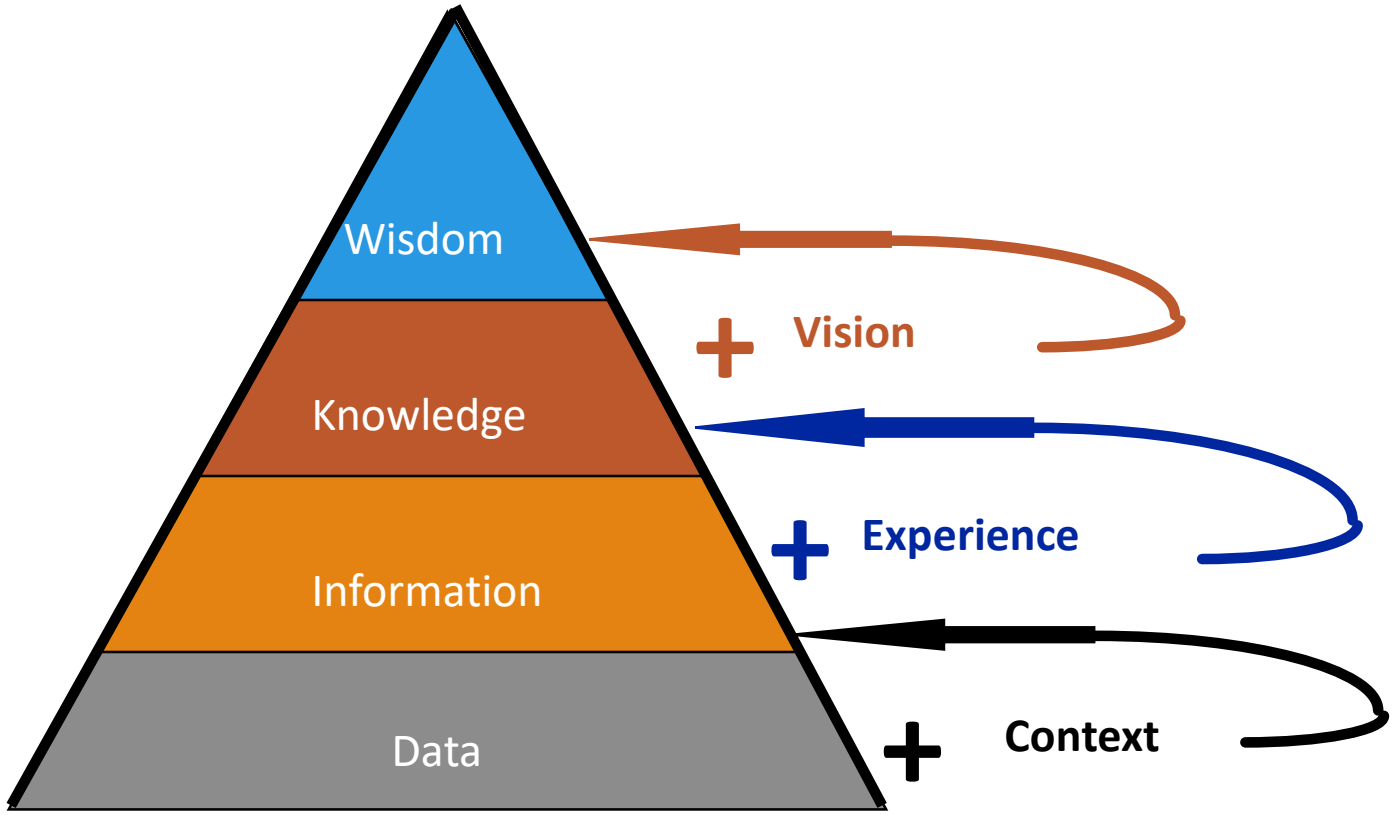
Knowledge Engineer

- a person who designs, develops, and implements expert systems (or other artificial intelligent applications).

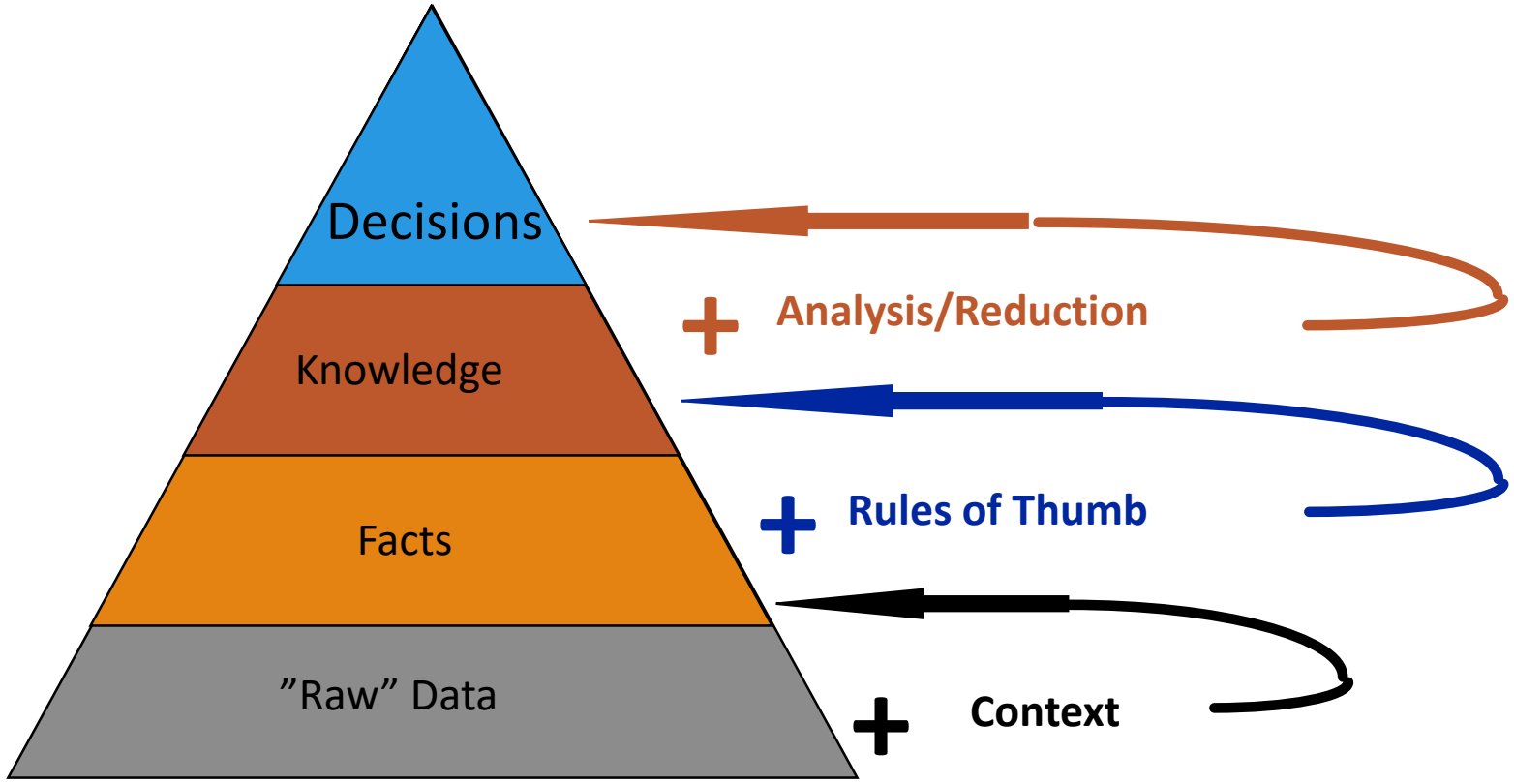
Phases in Expert System Development



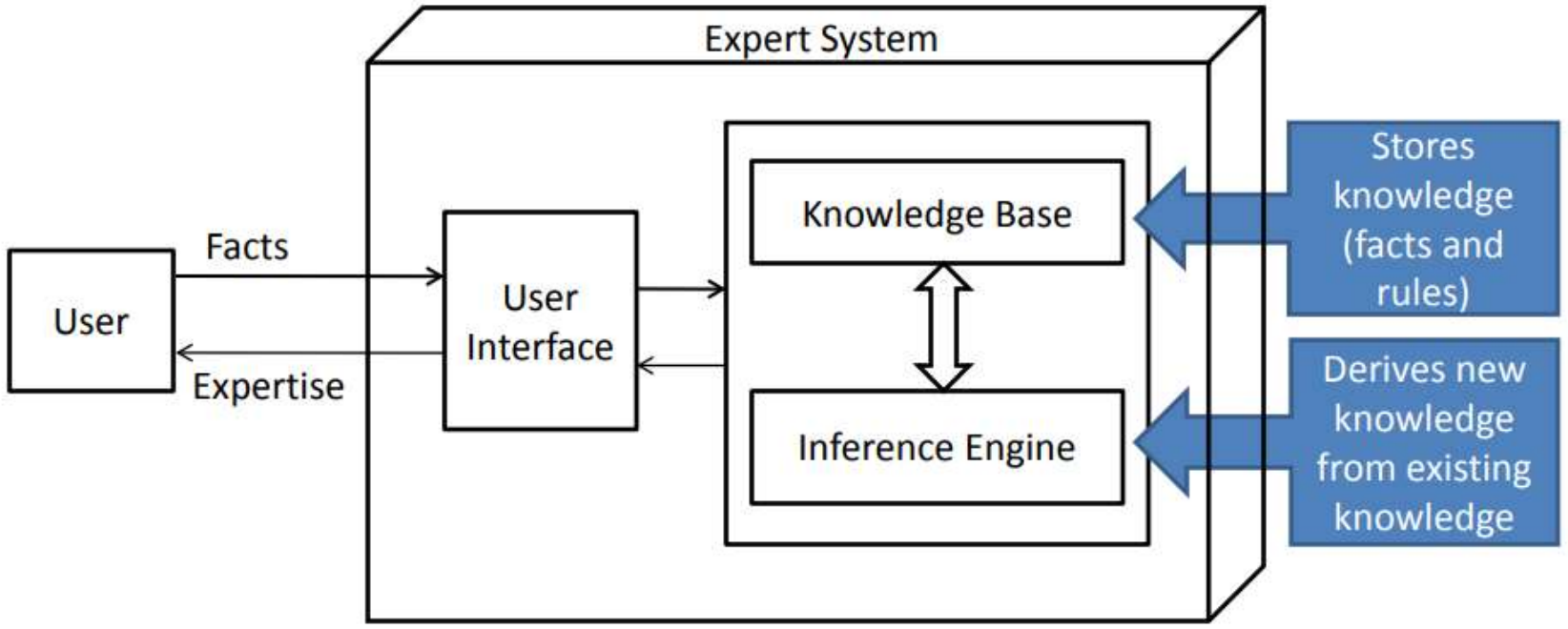
A Hierarchical Model of Intelligence



A Knowledge Engineer's View of Intelligence



Components of an Expert System



Components of an Expert System

- Inference Engine
  - Reasons about knowledge
  - Explains its reasoning
  - Helps programmer build correct/consistent expert systems
- Knowledge Base
  - Base is represented in easy to read English sentences
  - Order of rules is not critical
- The expert system shell (EXSYS) contains an existing inference engine
- The knowledge base is the only part of the system the programmer needs to develop

Knowledge Base

Contains the domain-specific, problem solving knowledge. There are two basic approaches.

- Rule: If-then statements that attempt to replicate the thought process used by the expert. (Known as Rule-Based Reasoning)
- Cases: Collection of previous solutions for situations and generalizes a solution for current situation (Known as Case-Based Reasoning)

Note: there are other ways to represent knowledge

Composition of a Rule

IF

Conditions

THEN

Conditions and Choices

ELSE

Conditions and Choices

NOTE:_____

REFERENCE_____

IF

IF part is simply a series of test conditions, expressed as English sentences or algebraic expressions:

Ex: Today is Wednesday or [X] > 1

The first rule will only be true if today is Wednesday.
The second rule will only be true if the variable X > 1.

Test is made by information:

- provided by the user
- derived by other rules
- obtained from other sources

Tests can be derived with other boolean operators, i.e. AND or OR

THEN

Contains conditions similar to if statements, but they are not tests.

They are statements of fact.

IF “today is Wednesday” may or may not be true.

Then “today is Wednesday” is a valid fact if the IF conditions are true.

The valid fact is then added to the knowledge base within the system.

THEN condition may also contain statements that assign a value or assigned value to a numeric or string value.

ELSE

ELSE part is the same as the THEN part and is applied if any of the IF conditions are FALSE.

ELSE part is optional and not needed in most rules.

Example of Rule-Based Reasoning

For example, an expert system might help a doctor choose the correct diagnosis based on a cluster of symptoms, or select tactical moves to play a game.

Example of Case-Based Reasoning

An auto mechanic who fixes an engine by recalling another car that exhibited similar symptoms is using case-based reasoning.

A lawyer who advocates a particular outcome in a trial based on legal precedents or a judge who creates case law is using case-based reasoning.

Reasoning Within an ES

When to use Rule-Based Reasoning?

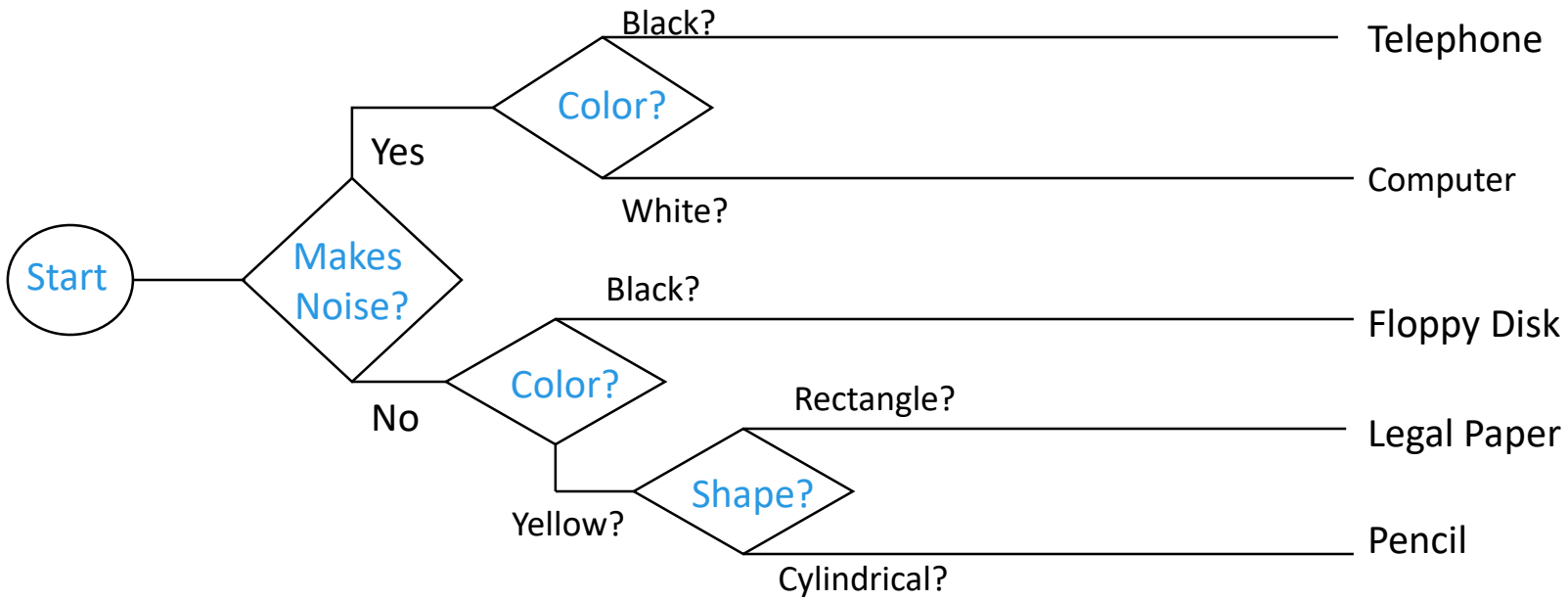
- When there is a lot of specific expert knowledge on a particular subject and the expert can solve the problem sequentially.
- When an explanation or an audit trail of the solution is required.

When to use Case-Based Reasoning?

- When the user wants to browse similar cases.
- When you have lots of typical situations or cases for the knowledge base.

Tree Structured Rules

Model: items of a desk to include a telephone, a computer, a pencil, legal paper, and floppy disks.



Converting Tree Structure to an Expert System

Rule 1: IF the item makes noise AND the color is black
- THEN telephone - Confidence 9/10

Rule 2: IF the item makes noise AND the color is white
- THEN computer - confidence 10/10

Rule 3: IF the item does not make noise AND the color is black
- THEN floppy disk - confidence 10/10

Rule 4: IF the item does not make noise AND the color is yellow AND the shape is rectangular
- THEN legal paper - confidence 7/10

Rule 5: IF the item does not make noise AND the color is yellow AND the shape is cylindrical
- THEN pencil - confidence 8/10

Confidence Modes

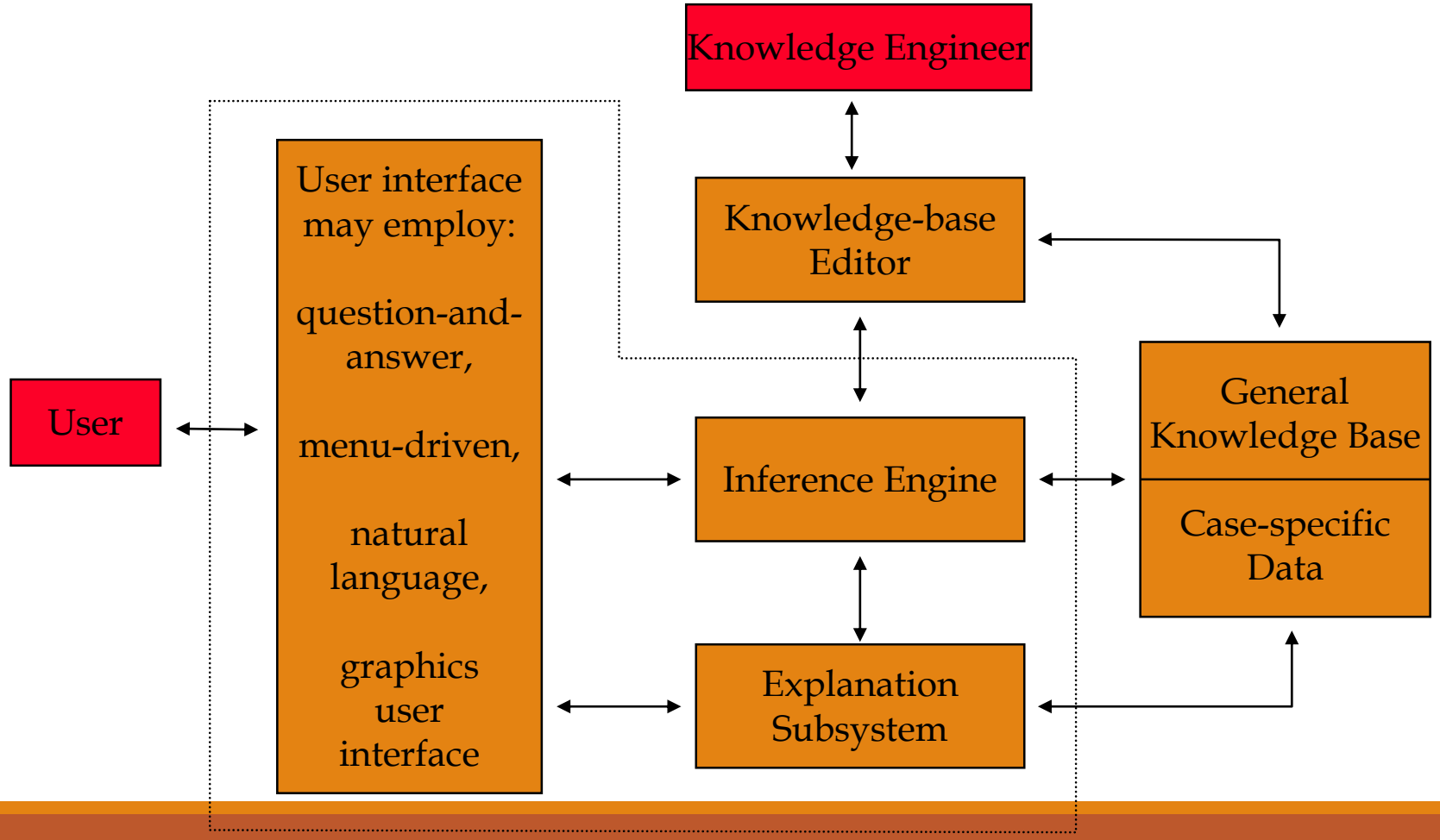
- One of the most powerful parts of an expert system is being able to create rules which state that an answer is probably, but not definitely, true.

- Process accomplished by assigning confidence modes

- Rule of Thumb -- Use the simplest one that will do the job!

- Ways to assign confidence modes:
  - 0-10
  - -100 to 100
  - Increment/decrement system
  - Custom formula

Architecture of Expert Systems



Inference Engine Control Strategies

Control Strategies for Execution of Rules

- Backward Chaining
  - Begins with a goal and works backwards towards the initial conditions
- Forward Chaining
  - Starts with available information and then draws conclusions

Which Strategy to Use?

- **When to use backward chaining?**
  - When a specific conclusion or set of conclusions is being sought or tested.
  - Common use: Classification
  - Ex. Which ***one*** of "these" cars should I buy?

- **When to use forward chaining?**
  - When you want all the possible conclusions.
  - Common use: Diagnosis
  - Ex. What job ***positions*** can I apply for?

Backward Chaining

Runs rules in a “goal-driven” way.

If a piece of information is needed, the program will automatically check all of the rules to see if there is a rule that could provide the needed information.

The program will then “chain” to the new rule before completing the first rule.

The new rule may require information that can be found in yet another rule.

Logic of why the information is needed goes backward through the chain of rules.

Backward Chaining Example

The program needs to know if the day is hot. Program will automatically check all rules to see if there is a rule that tells if the day is hot.

Rule 1:
IF
The day is hot
THEN
Go to the beach

Rule 35:
IF
It is summer
and It is sunny
THEN
The day is hot.

If there are rules in the system that tests if it is summer or that it is sunny, it will test those rules before rule 35 and test rule 35 before rule 1. The chain will continue until all applicable rules are tested.

Forward Chaining

Data-Driven way to run the rules.

In pure forward chaining rules are simply tested in the order they occur based on available data.

If information is needed, other rules are NOT invoked.

Instead, the user is asked for the information.

Backward chaining systems are not dependent on order, forward chaining rules are.

Many systems are a hybrid of the two.

Meta Rules

- Meta rules (also called control strategies) are rules about how to apply a strategy.
- A rule-based system consists of layers of rules
- Meta rules can determine which criteria a set of rules should be instantiated against.

- Ex: Two rules meet the firing criteria but result in different outcomes, which rule do you fire:
  - Meta rule could fire the rule that is most recent
  - Meta rule could fire the rule that is more specific

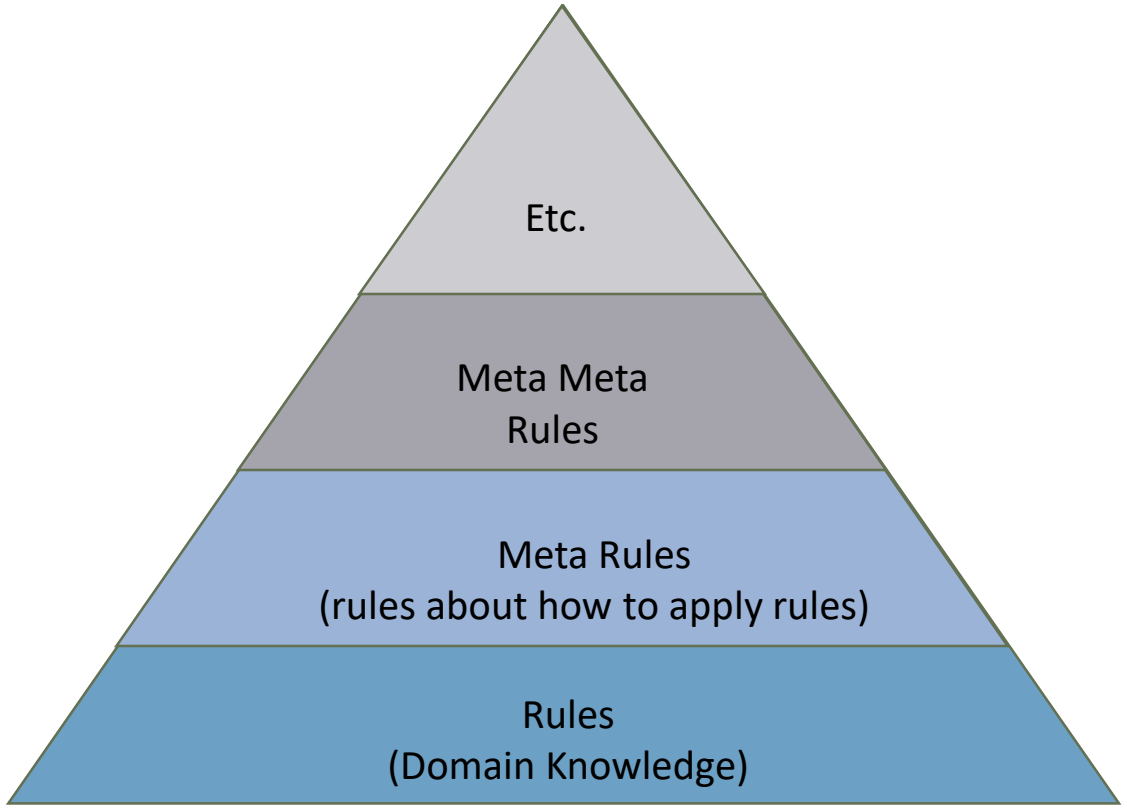
Example of Control Strategy

- Rules: If credit history is less than one year, then applicant is a high credit risk.
  - If applicant is a doctor, then applicant is a low credit risk.
- Facts: Cash is 25 years old
  - Cash has 6 months of credit history
  - Cash has been unemployed for three-fifths of a year
  - Cash is a doctor
- Meta Rule: If more than one rule meets the criteria then fire the rule that meets the most recent fact.
- Cash is a doctor is most recent fact, therefore, applicant is a low credit risk.

Example of a Meta-Rule

IF The car will not start
AND The electrical system is operating normally
THEN Use rules concerning the fuel system

Meta- Rule Hierarchy



Building an Expert System

- Analysis
- Specification
- Development
- Deployment

Analysis

User/expert: Identify a potential application.

Knowledge Engineer: Is expert system the answer?

- Task is well understood
- Expertise exists, is reliable, and the solution is generally agreed upon
- Task is not too hard (but not too easy, either)

Specification

User/Expert and Knowledge Engineer work together to define the objectives of the expert system application:

- Inputs
- Outputs
- Methodology

Development

Knowledge Engineer: Learns how the expert performs task:

- Called “knowledge acquisition”.
- Must cover current, historical, and hypothetical cases.

Develop a conceptual model of the ES.

- Framework consists of high-level descriptions of the tasks and situations.

Development (contd)

Decide how the inference, representation, and control structure can be used to replicate the decision process.

Build the knowledge base.

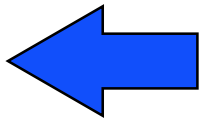
Verify and validate.

- Am I building the product right?
- Am I a building the right product?

Deployment

Install the system for routine use.

Fix bugs, update, and enhance.



Go back to development phase.

This loop remains active throughout the life cycle of the project.

Thank You