CHAPTER 5

Knowledge Engineering: Techniques & Tools

5.1. Introduction

Knowledge acts like a machine that takes in data and information at one end and spurts out decisions and actions at the other end. This idea can be put into a definition depicted in fig. 5.1.

Knowledge is the	ability skill expertise	to	manipulate transform create	data information ideas	to	perform skillfully make decisions solve problems

Fig. 5.1. Knowledge.

Creation of an ensemble of knowledge handling capabilities for problem-solving through computers is referred to as Knowledge Engineering (KE). KE is the process of acquiring, representing, organising, encoding, storing, processing and applying knowledge through computers. Its purpose is knowledge-based problem-solving in specified domains. KE, used mainly with reference to computer science, has been defined by Feigenbaum [1] as the process of reducing a large body of knowledge to a precise set of facts and rules (Table 5.2). The concept has come into widespread use in the '80s following the development and implementation of a new kind of computer programs (Expert Systems) aiming at embodying and communicating in a structured organized way the expertise related to specific domains of knowledge.

Developing knowledge-based application creates difficulties to knowledge engineers [2]. Knowledge-based system cannot be handled by general software engineering methodology. The lifecycle of knowledge based application and software application is different in many aspects. In order to achieve the objective of knowledge engineering, Knowledge-Based Engineering (KBE) application lifecycle [3] focuses on these six critical phases as shown in figure 5.2.

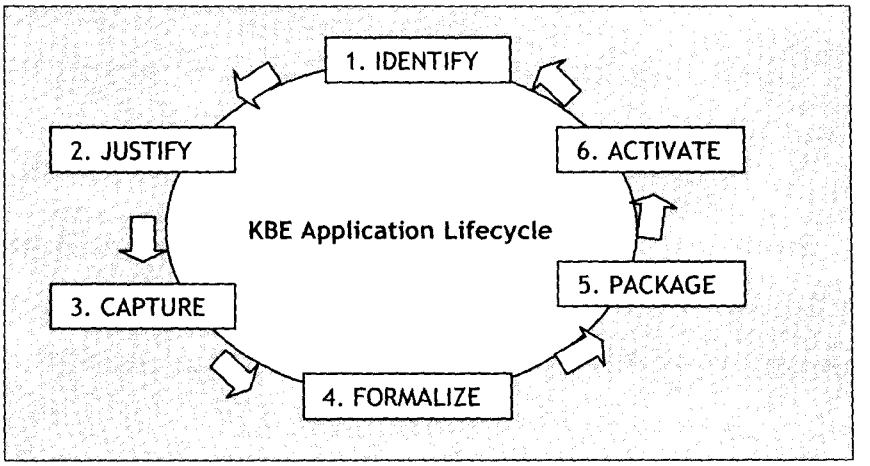

Fig. 5.2. Knowledge based engineering application lifecycle.

Expert Systems (ES) require the use of domain related 'expert' knowledge for their solution. The expert knowledge may be in the form of facts, relationships, procedures, heuristics, beliefs, experience and insights. It may variously be formal and categorical, quantitative and qualitative, incomplete and imprecise, uncertain and judgmental. Knowledge may be collected from many sources. A representative list of sources includes domain experts, books, computer databases, maps, flow diagrams, pictures, web-sites etc. These sources can be categorised into two types: documented and undocumented sources.

Domain experts are generally considered as the primary sources of knowledge for an expert system development. Experts should have developed domain expertise by task performance over a long period of time. One of the objectives of the knowledge acquisition is to find the experts' heuristics related to the task. Project experts should have enough experience to have been able to develop the domain insights that result in these heuristics.

Experts should be capable of communicating their knowledge, judgments, and experience and the methods they use to apply these to the particular task. Experts' temperament, cooperativeness, and working relation with the project team can have a major impact on the success and the speed of the knowledge acquisition.

This knowledge has to be put into an objective form for the knowledge base. The proper selection and design of a suitable knowledge representation scheme should be in tune with the requirements of the application domain. In addition, the proper selection should also depend on certain important properties of a scheme like expressive power and adequacy in context to the application

domain. In this chapter, we have tried to analyse some of these issues from the viewpoint of an expert system designer.

Section 5.2 contains the description of levels of knowledge. Knowledge categories are presented in section 5.3. In section 5.4, we describe the knowledge acquisition. Methods of knowledge acquisition are presented in section 5.5. Knowledge acquisition problems and possible ways of overcoming them are discussed in section 5.6. Section 5.7 contains the knowledge acquisition in the context of present work. Section 5.8 will be devoted to describe some knowledge representation schemes from the literature. Section 5.9 contains the object-oriented presentation in expert systems. In section 5.10, we analysed the relative suitability of such schemes as described in section 5.8. Lastly, some discussions are provided.

5.2. Levels of knowledge

There are two extreme levels of knowledge - shallow knowledge and deep knowledge. Shallow knowledge is the surface level information that can be used to deal with very specific situations. Deep knowledge refers to the internal and casual structure of a system and considers the interactions among the system components. Deep knowledge can be applied to different tasks and different situations. It is based on a completely integrated, cohesive body of human consciousness that includes emotions, common sense, intuition etc.

5.3. Knowledge categories

Knowledge can be differentiated into various categories - such as declarative knowledge, procedural knowledge, semantic knowledge, episodic knowledge and meta-knowledge.

Declarative knowledge

Descriptive representation of knowledge is a declarative knowledge. It is expressed in a factual statement. Declarative knowledge is especially important in the initial stage of knowledge acquisition.

Procedural knowledge

It includes step-by-step sequences and how-to types of instructions, it may also include explanations.

Semantic knowledge

Semantic knowledge reflects cognitive structure that involves the use of the long term memory.

Episodic knowledge

Episodic knowledge is autobiographical, experimental information organized as a case or an episode.

Meta-knowledge

Meta-knowledge means knowledge about knowledge. In AI, meta-knowledge refers to the knowledge about the operation of knowledge based systems i.e., about its reasoning capabilities.

5.4. Knowledge acquisition

Knowledge acquisition is the process by which expert system developers get the knowledge that domain experts use to perform the task of problem solving. This knowledge is then implemented to form the ES programs. Therefore, after domain selection, knowledge acquisition is very likely the most important task in an ES development. Acquisition of domain knowledge is a crucial phase of any knowledge engineering application and its weakness or failure can decisively block any future progress toward building an ES. Knowledge acquisition has to be systematic and comprehensive. It is desirable that, the knowledge must be accurate, complete in the sense that all essential facts and rules are included, free of inconsistencies and so on.

Table 5.1. Knowledge Processes [4,5 & 6].

Knowledge process by	Knowledge tasks
Liebowitz	1. Transform information to knowledge 2. Identify and verify knowledge 3. Capture and secure knowledge 4. Organise knowledge 5. Retrieve and apply knowledge 6. Combine knowledge 7. Create knowledge 8. Distribute/Sell knowledge
Wiig	1. Creation and sourcing 2. Compilation and transformation 3. Dissemination 4. Application and value realization
Van der spek	1. Developing new knowledge 2. Securing new and existing knowledge 3. Distributing knowledge 4. Combining available knowledge
Ruggles	1. Generation consisting of creation, acquisition, synthesis, fusion, adaptation 2. Codification consisting of capture and representation 3. Transfer
Staab et al.	1. Creation and import 2. Capture 3. Retrieve/Access 4. Use
Weber & Aha	1. Collect 2. Verify 3. Store 4. Disseminate 5. Reuse

5.4.1. Sources of knowledge

There are various sources of knowledge. We may classify them into two broad categories: (i) classical sources, and (ii) more recently available web-based sources.

Classical sources

A representative list of classical sources includes (i) domain experts, (ii) books and literature, films, computer databases, pictures, maps, flow diagram, etc. and (iii) real field case studies. Furthermore, these sources can be divided into two types: documented and undocumented knowledge. Undocumented knowledge resides in people's mind. Worthwhile to mention that in medical domain there are scopes of accumulating undocumented knowledge as gathered by the medical practitioners during their consultation with the patient. In this respect, domain experts might be considered as a good source of knowledge. Although there is a need for better methods of knowledge acquisition, including techniques to automate the process, but for the foreseeable future, most of the knowledge for any practical expert system in a complex domain will be obtained through the interaction of knowledge engineers and domain experts [7].

Web-based sources

Knowledge acquisition has got a new dimension in the recent years after the introduction of Internet and World-Wide-Web (WWW). WWW has not only revolutionized the dissemination process of information but also it has created novel opportunities for sharing literal knowledge via Internet. Peoples are now getting acquaintance with web-based computer technologies. Knowledge engineers, working in their fields of interest can search and access the relevant web documents as a source of knowledge.

5.5. Methods of knowledge acquisition

The elicitation of knowledge from the expert can be done by various ways. We may classify the methods of knowledge acquisition in three categories: manual, semiautomatic and automatic.

Manual methods

Manual methods are typically based on some kind of interview. The knowledge engineer elicits knowledge from the domain expert and / or other sources and then codes it in the knowledge base. The three major manual methods are interviewing (structured, unstructured, semi-structured), tracking the reasoning process, and observation.

Interviewing

Structured interview

Structured interview is a systematic goal-oriented interview. It establishes an organized communication between the expert and the knowledge engineer. Attentions to a number of procedural issues are necessary for structuring an interview. During the knowledge acquisition session the knowledge engineer should identify target question to be asked. Sample question, questioning techniques, question(s) type and level should be written prior to structured interview. Knowledge engineer should follow the guide lines for conducting interviews.

Unstructured interview

To start with, informal interviews are conducted for many knowledge acquisition requirements. It helps to get quickly to the basic structure of the domain and saves time. According to McGraw and Harboson-Briggs [8], unstructured interviewing seldom provides complete or well organized descriptions of cognitive processes. They pointed out different problems related to unstructured interviews. But, however, sometimes this kind of unstructured interviews should help the knowledge engineers.

Semi-structured interview

Semi-structured interviews are obviously a compromise between structured approach and unstructured approach. This approach is required when complete unfolding of the complexity of the problem domain is not possible.

Tracking the reasoning process

This refers to a set of techniques that attempts to track the reasoning process of an expert. Cognitive psychologists use this technique in discovering the expert's 'train of thought' while he / she reaches a conclusion.

Observations

Sometimes, it may be possible to observe the expert at work and thereby one can acquire knowledge.

5.5.2. Semi-automatic methods

Methods intended to help the knowledge engineers by allowing them to execute the necessary tasks in a more efficient and / or effective manner and also intended to support the experts by allowing them to build knowledge bases with little or no help from knowledge engineers are semiautomatic methods.

5.5.3. Automatic methods

In this method the role of the expert is minimal (limited to validation) and there is no need for a knowledge engineer. For example, the induction method can be administered by any builder (e.g., a system analyst).

5.6. Problems in knowledge acquisition

There are some problems with knowledge acquisition [9] mainly concentrating on two aspects: (i) availability of source(s) and (ii) transferring knowledge. To overcome these problems many efforts have been made [10]. For example, researches on knowledge acquisition tools are going on [11] to focus on ways to decrease the representation mismatch between the human expert and the program under development. Several expert system development software packages such as EXSYS, Level5 and VP expert greatly simplify the syntax of the rules (in a rule-based system) to make them easier for an expert system builder to create and understand without special training. Also, a natural language processor can be used to translate knowledge.

In case of web based acquisition, any search engine results a large number of directories. It is sometimes very difficult to have a comprehensive list of the proper web sites of domain interest. Although one may get some good starting places. Some problems associated with web based knowledge acquisition are:

a) There is a big question of reliability of information as there is lack of quality control at stage of production and uploading.

b) It is possible to read a web page without having seen context pages or the cover page containing disclaimers, warnings etc.

c) Authors of web pages, news articles, e-mails, etc., sometimes remain unidentified.

5.7. Knowledge acquisition in the context of present work

As primary sources, we used human experts, books, manuals or other written materials discussing the domain. As the problem domain is concerned with fever management and resuscitation of child, so at the very initial stage of this work, the knowledge of domain parameters was elicited from various books on the subject, published papers and workshop reports. Major parts of the documents were collected from some well-known medical institutes' websites, having experience and contribution in providing the research and development in peadiatric domain. We had also taken help from doctors.

The domain terminology, categories and jargon were extracted from the above said literatures and a glossary of pertinent domain terms was prepared. It provided a benefit on the subsequent steps of knowledge engineering.

The steps involved in new-born resuscitation management and child fever management were identified on the basis of documents acquired from different sources. The possible set of precautions required for each steps were verified by the domain experts. The outcomes were studied from relevant text books and publications and domain experts.

Domain experts are one of the primary sources of knowledge for an ES. Knowledge acquisition primarily refers to the process of eliciting the needed knowledge from domain experts. Knowledge acquisition is concerned with eliciting the facts, rules, patterns, heuristics, and operations used by experts to solve problems in a particular domain. In the present work, as the primary source of knowledge, three domain experts having 20 to 30 years of experience

had been consulted through structured interviews. Forms were prepared to record the knowledge extracted from those experts. The experts were requested to give their judgments for different set of possible observations. The representative sources of domain experts used in this work are:

(i) Dr (Mrs.) Mridula Chatterjee, Pediatrician, North Bengal Medical College and Hospital, Sushrut Nagar, West Bengal, India.
(ii) Dr. S. S. Debnath, (O & G), C.M.O., P.M.Hospital, Visva-Bharati, Santiniketan, West Bengal, India.
(iii) Dr. M. G. Goswami, Senior Medical Officer, Siliguri, West Bengal, India.

5.8. Knowledge representation methods

The way a knowledge engineer models the facts and relationships of the domain knowledge is called knowledge representation. The two types of knowledge that need to be represented in a computer are declarative knowledge and procedural knowledge. Declarative knowledge signifies facts about objects, events, and about how they relate to each other and procedural knowledge signifies the way to use the declarative knowledge.

Several common knowledge representation schemes have been discussed in the literature [9, 12-16] including logic, semantic networks, production rules, frames, scripts, and OAV-triplets as classical methods; and the relatively new paradigm: object-oriented (O-O) approach.

5.8.1. Logic

Logic is a formal method of reasoning. The application of logic as a practical means of representing and manipulating knowledge in a computer was not demonstrated until the early 1960s [17]. Many concepts which can be verbalized can be translated into symbolic representations. These symbolic structures can then be manipulated in programs to deduce various facts to carry out a form of automated reasoning. The logic process takes in some information called premises and produces some outputs called conclusions. Logic is basically classified into two categories: propositional logic and predicate logic (first order predicate logic).

Propositional logic (PL) is the simplest form of logic. Here all statements made are called propositions. A proposition in propositional logic takes only two

values i.e. either the proposition is 'TRUE' or it is 'FALSE'. Consider the following statements:

Statement 1: Sun rises in the East.

Statement 2: Diamond is a brittle material

Both these statements are propositions, with the former having a value of TRUE and the later having a value of FALSE.

Valid statements or sentences in PL are determined according to the rules of propositional syntax. This syntax governs the combination of basic building blocks such as propositions and logical connectives. There are two kinds of propositions: atomic propositions (simple propositions) and molecular propositions (compound propositions). Molecular propositions are formed by combining two or more atomic propositions using a set of logical connectives 'not', 'and', 'or', 'if', 'then', and 'if and only if'.

The inference rules of PL provide the means to perform logical proofs or deductions. A key inference rule in PL is modus ponens.

Modus ponens: From P and $P \rightarrow Q$ infer Q

or

$$\frac{P \qquad P \rightarrow Q}{Q}$$

For example,

given: (Maya is a mother)
and (Maya is a mother) $\rightarrow$ (Maya has a child)
conclude: (Maya has a child).

PL works well in situations where the result is either TRUE or FALSE, but not both. However, there are many real life situations that can not be treated this way.

First Order Predicate Logic (FOPL) or predicate calculus has played one of the most important roles in AI for the representation of knowledge. In FOPL, statements from a natural language like English are translated into symbolic structures comprised of predicates, functions, variables, constants, qualifiers, and logical connectives. The symbols form the basic building blocks for the

knowledge and their combinations into valid structures are accomplished using the syntax for FOPL. Once the structures have been created to represent basic facts or procedures or other types of knowledge, inference rules may then be applied to compare, combine, and transform these 'assumed' structures into new 'deduced' structures. This is how automated reasoning or inferencing is performed.

Like propositional logic, a key inference rule in FOPL is modus ponens. From the assertion 'Tom is a cat' and the implication 'all cats are good' we can conclude that 'Tom is good'.

Assertion: Cat (Tom)
Implication: $\forall x$ cat(x) $\rightarrow$ good(x)
Conclusion: good (Tom).

In general, if 'a' has property P and all objects that have property P also have property Q, then the conclusion is a has property Q.

Modus ponens:

$$\frac{P(a) \qquad \forall x\, P(x) \rightarrow Q(x)}{Q(a)}$$

5.8.2. Semantic networks

A graphic notation for representing knowledge in patterns of interconnected nodes and arcs is called a semantic network. Computer implementations of semantic networks were first introduced by Quillian [18] to model the semantics of English sentences and words.

What is common to all semantic networks is a declarative graphic representation that can be used either to represent knowledge or to support automated systems for reasoning about knowledge. The most common kinds of semantic networks [19] are shown below:

- Definitional networks
- Assertional networks
- Implicational networks
- Executable networks
- Learning networks
- Hybrid networks

Semantic nets provide a more natural way to map to and from natural language than do other representation schemes. Network representations give a pictorial representation of objects, their attributes and the relationship that exist between them and other entities. The following rules about nodes and arcs generally apply to most of the semantic nets:

a) Nodes in the net represent either entities or attributes or states or events.
b) Arcs in the net give the relationship between the nodes and labels on the arc specify what type of relationship actually exists.

For example, a class of objects known as 'Bird' is depicted in fig. 5.3. The class has some properties and a specific number of class named 'Tweety' is shown. The colour of 'Tweety' is seen to be yellow.

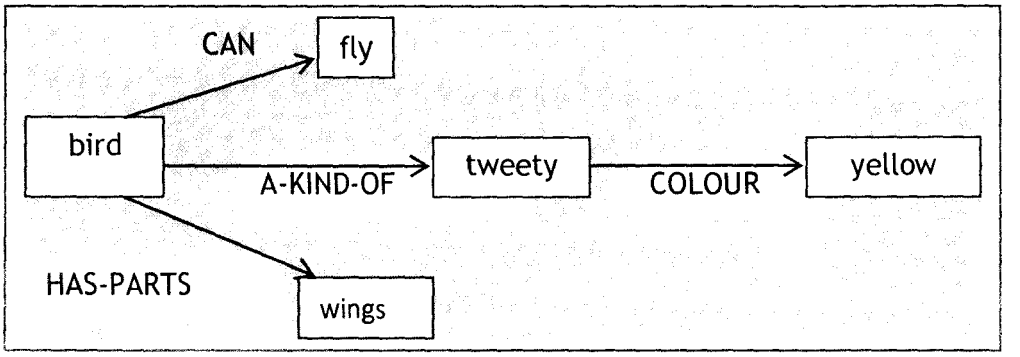

Fig. 5.3. Fragment of a semantic net

5.8.3. Rules

Rules provide a formal way of representing recommendations, directives, or strategies; they are often appropriate when the domain knowledge results from empirical associations developed through years of experience solving problems in an area. A rule has two parts. The first part is a premise of conditions connected by logical-AND or logical-OR relationships. The second part is a conclusion. When the premise of a rule is true, the conclusion of the rule will become true. In some systems rules may be implemented by semantic networks or OAV, as in MYCIN [20], the medical diagnostic system developed at Stanford University. Alternatively, rules may be represented by frames, as in IntelliCorp's knowledge engineering environment [21].

Table 5.2. Characteristics of rule representation.

	First Part	Second part
Names	Premise Antecedent Situation IF	Conclusion Consequence Action THEN
Nature	Conditions Similar to Declarative Knowledge	Resolutions Similar to Procedural Knowledge
Size	Can have many Ifs	Usually has one conclusion
Statement	AND statements OR statements	All condition must be true for a conclusion to be true If any of OR statement condition is TRUE the conclusion is TRUE

In expert systems jargon the term rule has a much narrower meaning than it does in ordinary language. It refers to the most popular type of knowledge representation technique, the rule-based representation. Rules are expressed as IF-THEN statements, as shown below [22]:

Rule 1. If a flammable liquid was spilled,
called the fire department.

Rule 2. If the pH of the spill is less than 6,
the spill material is an acid.

Rule 3. If the spill material is an acid,
and the spill smells like vinegar,
the spill material is acetic acid.

Fig.5.4. Rule-based representation using statement.

These are rules that might exist in a crisis management expert system for containing oil and chemical spills. Rules are sometimes written with arrows (→) to indicate the IF and THEN portions of the rules.

Rule 2 in this notation would look like:

If the pH of the spill → the spill material
is less than 6 is an acid.

Fig.5.5. Rule-based representation using arrows.

In a rule-based expert system, the domain knowledge is represented as sets of rules that are checked against a collection of facts or knowledge about the current situation. When the IF portion of a rule is satisfied by the facts, the action specified by the THEN portion is performed. When this happens, the rule is said to fire or execute. A rule interpreter compares the IF portions of rules with facts and executes the rule whose IF portion matches the facts.

There are two important ways in which rules can be used in a rule-based expert system: forward chaining and backward chaining.

Forward chaining is a 'data-driven' approach. In this approach one starts from available information as it comes in, or from a basic idea, then to draw conclusions. Backward chaining is a 'goal-driven' approach in which one starts from an expectation of what is to happen (hypothesis), then seek evidence that supports (or contradicts) his/her expectation. The inference chain created by backward chaining is identical to the one created by forward chaining; but however, the order and actual number of states searched can differ. The preferred strategy is determined by the properties of the problem itself.

5.8.4. Frames

The concept of frames as a means of representing knowledge was introduced by M. Minsky [23]. He formulated the notion of frames like data structures for presenting of stereotypical situations [24]. The idea is that the whole data describing a given situation has to be collected in one place - the frame of the situation. As is known, frames have a common part and slots which represent the different characteristics of the object or the relationships with other objects. Important constituent part of every frame is a pointer to its parent frame. In this way, a hierarchy with properties inheritance is defined. Slots are filled by fillers, which can be atomic values and names of other frames, too. Frames, which compose the model of the knowledge domain, are called generic frames or type frames and frames which describe a particular object known as individual frames [25]. A distinctive feature of inheritance in frame systems is that it is defeasible, i.e. the property is inherited only if in a specialization sub-frame or in an instance of the same frame, it is not replaced by another value explicitly. Another aspect of reasoning with frames is the classification - putting of a frame to its exact place in the hierarchical structure. Classification and inheritance do not represent the full scope of possibilities of deductive reasoning; they do not have the computational power of a programming system, but they do support knowledge engineering efficiently [26].

A disadvantage of a frame-based knowledge base is that it cannot work with objects, whose characteristics are not known in advance and it cannot process untypical situations [27]. Another disadvantage is that like in first order predicate logic, the knowledge domain has to be static. Change, if it occurs, is an exception rather than a rule. The third disadvantage is connected with procedural knowledge in frame systems. The procedural knowledge is not represented by a frame but with programming code. The system can perform reasoning with this knowledge, but not about it [28].

Frame-like structures, along with rules, are exercised in expert systems [29, 30 & 31] provide examples of tool and shell developments which facilitate the manipulation of knowledge presented by frames. Frames are a convenient method for modeling of the knowledge domain in intelligent systems for planning. Even though Alshawi et al. [32] used kind of predicate logic for knowledge representation, the connection with the data base goes through 'conceptual predicates' which have much in common with the concept of frames. Frames with their capabilities for easy representation of concepts with high level of abstraction are a proper method for knowledge representation when the goal is realization of a natural language sentences analyzer.

Frame representations have become popular enough that special high level frame-based representation languages have been developed. Most of the languages use LISP as the host language. Several frame languages have now been developed to aid in building frame based systems. They include the Frame Representation Language (FRL) [33], Knowledge Representation Language (KRL) [34], and KLONE [35].

5.8.5. Scripts

Scripts were proposed by Roger Schank [36] at Yale University. Scripts are composed of a series of slots that describe, in sequence, the events that we expect to take place in familiar situations. Just as the concept of frames is based on the assumption that we have a set of expectations about objects, the use of scripts assumes that we also expect certain sequences of events to occur in particular times and places.

Reasoning in a script begins with the creation of a partially filled script situation to meet the current situation. Next a known script which matches the current situation is recalled from memory. The script name, pre-conditions or other key words provide index values with which to search for the appropriate script.

Inference is accomplished by filling in slots with inherited and default values that satisfy certain conditions.

Scripts have now been used in a number of language understanding systems, such as SAM, PAM, POLITICS, FRUMP, IPP, BORIS etc. at Yale University by Shank and his colleagues. They all used some form of script representation schemes.

5.8.6. Object-attribute-value triplets

The object-attribute-value triplets (OAV-triplets) provide a particularly convenient way in which to represent certain facts within a knowledge base and may be extended to provide the basis for the representation of heuristic rule. Each OAV-triplets is concerned with some specific entity or object. This method represents knowledge using three tuples: (O, A, V). O is the set of objects, which may be physical or conceptual entities. A represents the set of attributes characterizing the general properties associated with objects. V (values) specifies the nature of the attributes.

An expert system stores data about real-world entities. In knowledge representation theory, the real-world entities are objects. Each object has one or more attributes or properties, and the attributes have values; for example fig.5.6.,

Object	Attribute	Value
Bird	Colour	Green
John	Height	Tall

Fig.5.6. KR using OAV-triplets.

The object is Bird, the attribute is Colour, and the value is Green. Another example, the object is John, the attribute is Height, and the value is Tall.

OAV is more structured than a semantic network. However, when the number of objects increases, an OAV system becomes difficult to manage.

5.8.7. Object-Oriented methodology

An object is an independent entity represented by some data and a set of operations (methods and capabilities) [37]. So, an object can be used to represent a variety of knowledge. Knowledge (K) can be formally represented by three tuples, K = (C, I, A). C is a set of classes represented by class objects. I is a set of instances represented by instance objects. A is a set of attributes possessed by the classes and instances.

5.8.7.1. Classes

A class is a description of a group of similar instance objects [37]. Each class has a unique name and a set of attributes that define the properties of the class. A class may be a sub-class of another class and may inherit properties from its parent class as discussed in sub-section 5.8.7.4.

5.8.7.2. Instance objects

Instance objects are members of classes. Their properties are defined by their parent classes. Each instance object consists of three sets of attributes:

(a) Name - the name of the object, which is unique in the system. It is used to identify the object.

(b) Class - the class that contains the object.

(c) Instance attributes - attributes belonging to the instance object. Some operations may be performed on these attributes. The behaviour of the object is determined by the values of these attributes.

5.8.7.3. Attributes and methods / operations

The property of an attribute is determined by its type and value. The type of an attribute is defined by its class, while the value may be defined in its class or its instance object. A set of generic attributes can be associated with the every object in a class, say furniture, for example. All furniture have a cost, dimensions, weight, location, and colour among many possible attributes.

Methods are a kind of attribute belonging to objects. They are used to represent capabilities, not to store data, and are defined in classes. Methods cannot be modified during consultations.

5.8.7.4. Inheritance

Properties of a class can be inherited from its parent's class. This feature permits factoring knowledge into a class hierarchy. Thus, it encourages modular design of knowledge. The system adopts the inheritance rules, which are similar to those in smalltalk [38], as follows:

(a) If class A inherits from class B, then the objects of class A support all operations supported by objects of class B.

(b) If class A inherits from class B, then class A's attributes are a superset of class B's attributes.

Once the class has been defined, the attributes can be reused when new instances of the class are created. For example, assume that we were to define a new object called table that is a member of the class furniture. Table inherits all of the attributes of furniture [38].

Every object in the class furniture can be manipulated in a variety of ways. It can be bought and sold, physically modified, or moved from one place to another. Each of these operations will modify one or more attributes of the object.

The object chair (and all objects in general) encapsulates data (the attribute values that define the chair), operations (the actions that are applied to change the attributes of chair), other objects (composite objects can be defined [39]), constants (set values), and other related information. Encapsulation means that all of this information is packaged under one name and can be reused as one specification or program component.

5.8.8. Knowledge acquisition tools

The ready-made templates for acquiring domain knowledge from experts are the knowledge acquisition tools. The first step of is to select the target medical area and select experts to gain domain specific knowledge. The next step is then to transfer the knowledge into computer interpretable knowledge based on the designed knowledge representation schemes discussed above. This section summarise the comparisons (Table 5.3.) of some important acquisition tools on the main five aspects: (i) Established domain ontology, (ii) Abilities of handling uncertainties, (iii) Representation schemes, (iv) Specific knowledge representation language and (v) Guideline execution engine or inference engine.

Many knowledge acquisition tools have been developed for ES/DSS. Among them, some tools [40,41] are designed for acquiring medical domain specific knowledge, and others [42,43] are designed specially for the acquisition of medical guidelines which can be used as the best and standardized clinical procedures. Some other guideline-based medical DSSs which are frequently mentioned in the literature such as GLIF [44], EON [45] and Arden Syntax are not listed in the table, as they are represented by Protege and UMLS-based knowledge acquisition tool.

Table 5.3. Some medical knowledge acquisition tools comparisons [46].

	Features				
Knowledge Acquisition Tools	**Established domain ontology**	**Abilities of handling uncertainties**	**Representation schemes**	**Supported by a specific knowledge representation language**	**Need a guideline execution engine/inference engine**
PROforma	N/A	Use argumentation mechanism to make diagnostic and therapeutic decisions	Structured (Object-oriented representation: Plans) and rules	Yes	Yes (PROforma enactment engine)
GLARE	Yes	Use a threshold value to compare the diagnosis' score	Structured (Object-oriented representation: Actions)	Yes	Yes
Protege	Yes	Not mentioned	Structured (Frames)	Yes	Yes
AsbruView	N/A	Use 'time annotations' to represent uncertainty in starting time, ending time, and duration of a plan	Structured (Time-Oriented, Skeletal Plans)	Yes	Yes
UMLS-based knowledge acquisition tool	Yes	Not mentioned	Logic and rules (Medical Logic Modules (MLMs))	Not mentioned	Yes
CMDS: object-oriented knowledge acquisition editor	Yes	Use certainty factor or belief degree to model uncertainty	Rules	Not mentioned	Yes

Uncertainty handling is one of the major challenges for most knowledge acquisition tools and hence the following review is mainly focused on the comparison of the 'abilities of handling uncertainties' [46].

PRO*forma* [47] is a guideline acquisition tool that contains expressive constructs for describing uncertain aspects of a guideline [48]. Its ability to make decision under uncertainty is provided by means of argumentation mechanism. In this kind of argumentation mechanism, diagnostic and therapeutic decisions are defined in terms of a set of options, and the decisions are made by using argument rules and commitment rules. Argument rules can support or oppose a decision option, and eventually establish a preference order on the options. Commitment rules are used for the selection of a decision option based on the preference order that argument rules have established. However, if a decision can be made without uncertainty, then those argument rules can be ignored and the decision can be made solely by using the commitment rules [47].

GLARE [49] has restricted capability in handling uncertainties in diagnostic decisions. In GLARE, diagnostic decisions are represented as a set of triples: <diagnosis, parameter, score> (where, in turn, a parameter is a triple of <data, attribute, value>), a threshold value is used to compare each diagnosis' score, and all alternative diagnoses are shown to the user - a physician, together with their scores and the threshold value. GLARE lets the user to choose diagnostic decisions among those alternatives that the system provides in a list. A warning is given if the user chooses a diagnosis whose score does not exceed the threshold value.

AsbruView [50] can handle uncertainty in temporal scopes by 'time annotations' which is used to specify the temporal aspects of its structured representation format: time-oriented, skeletal plans. A time annotation specifies different points in time and duration in relation to a reference point in time, such as the earliest starting shift (ESS), the latest starting shift (LSS), the earliest finishing shift (EFS), the latest finishing shift (LFS), the minimal duration (MinDu) and the maximal duration (MaxDu). These data specify the temporal constraints within which an action must be taken or a condition must be satisfied in order to trigger an action. The time annotation allows a representation of uncertainty in starting time, ending time, and duration of a plan.

The UMLS-based knowledge acquisition tool does not take uncertainties into consideration during its design and development processes. The CMDS: object oriented knowledge acquisition editor has a better function of handling uncertainty, because it associates with each rule a certainty factor that represents how true the rule is. The certainty factor ranges from -1 to 1, where -1 means the rule is known to be false, 0 means no information is known, and 1 means the rule is known to be true. For example, after the user defines a rule called 'Rule1': 'IF the symptom pattern is S, THEN the disease is D', the system

would pop out a dialogue box asking the user to input rule settings about 'Rule1', and the settings include the certainty factor of the rule. Protégé deals little with uncertainty.

Except for PRO*forma* and AsbruView, those medical domain specific knowledge or clinical guideline acquisition tools listed in Table 5.3 have a common characteristic: the established domain ontology, such as a set of ready-made and configurable templates.
Although the ontology provides guided assistance for expert physicians to express their medical knowledge into a computer-interpretable format, it may also restrict some of their knowledge from being extracted. This is because in reality medical DSS developers may only have limited medical knowledge in a chosen domain and the developed ontology may not be able to cover all the formats required to express the domain knowledge.

5.9. Expert systems development and O-O technology

Although there is the lack of standard definition of what is the O-O approach, but however, the properties like encapsulation, inheritance, polymorphism are considered useful for O-O approach. O-O is now being exploited for analysis and design, and people are sharing their experiences.

The ideas behind object oriented programming (OOP) and object oriented technology (OOT) date back to the forties [51]. These ideas, however, were not put into practice until the introduction of the Simula_67 programming language [52]. Simula, a superset of Algol, was designed for describing a wide class of discrete event simulations and implementing them for simulations. Simula objects represent data and an operation on the data. These objects communicate with each other through messages to determine their next action. Although primitive by today's standards, Simula provided the first insight into the value of OOP.

One of the most important events during the eighties that spurred the interest in AI was the marketing of expert system development shells. Most of the early shells were rule-based. However, given the appeal of object-oriented systems, as demonstrated by Smalltalk's success, the demand pushed vendors to offer tools with object-oriented techniques. These tools, commonly called frame-based development programs (but sometimes called hybrid tools), usually combine object-oriented techniques with rule-based programming. New procedural languages with object-oriented techniques also surfaced, such as

Objective C, C++, Pascal Object, Modula-2 and Lisp extensions such as Scoops, Flavors, Loops and the Common Lisp Object System (CLOS).

A review of systems developed during the later eighties and early nineties clearly shows a swing toward object-oriented techniques [53]. This trend was due partly to the availability of relatively inexpensive frame-based shells that ran on a variety of platforms. Two of the earliest frame-based shells, the knowledge engineering environment from IntelliCorp and the automated reasoning tool from inference, offered AI researchers powerful tools, but were costly and ran on mainframes or workstations, preventing their widespread use. In the mid-eighties, vendors began marketing cheaper object tools, many of which ran on a PC. This situation led to the accelerated development of frame-based expert systems. Most important, it opened the door at most universities for teaching object-oriented technology (OOT) techniques to the next generation of AI researchers. Most corporations - including many in the Fortune 500 - are focusing on client-server and object-oriented problems. These organizations have come to recognize AI in general and OOT in specific, as a standard way of doing business.

The vendors found that although terms such as 'AI' and 'expert systems' might have fallen out of favour in some circles, their clients' still wanted the object-oriented capability of their products. Companies are using AI but not promoting it, and vendors marketing products with AI capabilities but not advertising it. The irony - even if the spotlight is no longer on AI, AI's contributions will continue to positively affect future information processing, only under other labels [54].

5.10. Analysing KR schemes

Flexibility is the major advantage of semantic networks. This flexibility also exists in object-oriented (O-O) knowledge representations where, by storing the names of their objects as the attributes of an instance object, relations between instance objects can be established dynamically. The object-oriented (O-O) construct can be viewed as dynamic semantic network. The is-a links of semantic networks can be implemented in object-oriented (O-O) representations by relationships between classes and sub-classes or between classes and instances. Has-a links can be implemented by the relationships between classes and attributes. Therefore, object-oriented knowledge representation has the same power as a semantic network but is much more structured.

Semantic networks, rules, and OAV representations are not structured enough. A significant increase in the number of objects or rules makes the system difficult to manage. When the value of an object or an attribute is modified, it is difficult to pinpoint the effects on the whole system. The encapsulation property and structuredness of object-oriented (O-O) knowledge representations give them a distinct edge over these three representations.

Frames are more structured than semantic networks, rules, and OAV representations, since related attributes and rules can be grouped into frames hierarchically. However, modularity of knowledge represented in frames cannot be clearly defined, and frame representation lacks flexibility. In object-oriented (O-O) knowledge representation, which is quite similar to frames, knowledge can be arranged in a hierarchical form using classes.

In the context of the present problem domain, we now analyse the relative suitability of different KR schemes. When the domain knowledge is vast and varied, the knowledge can become unmanageable. To handle a large knowledge base it is suggested [55] that the structuredness and modularity is necessary where knowledge is varied. A common disadvantage in Semantic Networks, Rules and OAV representations is that they are not structured enough [16]. It is very difficult to manage a system with these representations when the number of objects and rules increases significantly. According to some researchers [56], some applications such as engineering processes, manufacturing and communications are expected to contain 100,000 rules or more. It is then very difficult to pinpoint the effects on the whole system if a value of an object or an attribute is modified.

The properties like encapsulation and inheritance of O-O approach are really attractive for large, integrated information systems. The encapsulation property prevents object manipulation except by defined operations. Inheritance is a valuable mechanism which enhances reusability and maintainability of software. Because O-O approach minimizes object interdependency [57] the knowledge can be structured.

A common disadvantage in OAV triplets and rules is that there may be some redundancy in information which may lead to some inconsistency. There is no such redundancy problem with Semantic Networks, frames and O-O forms. Moreover, O-O approach to KR supports high level of knowledge abstraction, an important advantage over other classical approaches.

After the analysis, we prefer O-O representation to improve consistency, understandability, maintainability and modifiability of knowledge base. Last, but not least, in the evolution of an expert system [22], prototyping may have an adverse impact on modifiability and maintainability of knowledge bases since these may be patched and modified several times. This may, however, be overcome by the use of O-O approach. As the system grows, the major changes will be with the addition of new objects or deletion of old objects rather than modifying the old objects. In this respect O-O approach is considered very useful for rapid prototyping, an added advantage.

5.11. Some ES and ES development tools using different KR-schemes

Some expert systems and ES-developmental tools [22, 55, 56, & 58] with the kind of knowledge representation scheme(s) and control for knowledge based scanning are shown in table 5.4.

Table 5.4. Some ES/ES-tools using different KR-schemes.

ES/ES-tools	Representation	Control
ADVISOR II	Rule-based	Backward chaining
AI/RHEM	Rule-based	Forward chaining
ARBY	Rule-based	Backward chaining
ART	Rule-based, Frame-based	Forward and backward chaining
BABY	Rule-based	Forward chaining
CLOT	Rule-based	Backward chaining
CODES	Rule-based	Backward chaining
DELTA	Rule-based	Forward, backward chaining
DIPMETER ADVISOR	Rule-based	Forward chaining
DSCAS	Rule-based	Forward chaining
DUCK	Logic-based, Rule-based	Forward, backward chaining
EMYCIN	Rule-based	Restrictive backward chaining
ESE	Rule-based	Backward chaining
EXPERT	Rule-based	Forward chaining
EXSYS	Rule-based	Backward chaining
FAITH	Frame-based	Backward and forward chaining
FOLIO	Rule-based	Forward chaining
GOLDWORKS	Rules, Frames, Objects	Control over direction
GURU	Rule-based	Backward chaining, Limited forward chaining
GUSS/1	Rule-based	Backward and forward chaining
IMACS	Rule-based	Forward chaining
KES	Rule-based, Frame-based	Backward chaining

KES II	Rule-based, classes	Backward chaining, Limited forward chaining
LEONARDO	Rules, Frames, Procedures	Backward and forward chaining
LEVEL5 (OBJECT)	Large-hybrid-object-oriented, Rule-based	Forward and backward chaining
LISP	Procedure-oriented, functional, symbolic expressions	Forward, backward chaining
M.1	Rule-based	Backward chaining
MEDICO	Rule-based	Forward chaining
MES	Rule-based	Forward chaining
MI	Rule-based	Forward chaining
MUD	Rule-based	Forward chaining
MYCIN	Rule-based	Backward chaining
NEURAX	Rule-based	Forward, backward chaining
NEXPERT	Rule-based	Forward, backward chaining
ONCOCIN	Rule-based	Forward, backward chaining
OPS5	Rule-based	Forward chaining
PDS	Rule-based	Forward chaining
Plant/cd	Rule-based	Backward chaining
PROJCON	Rule-based	Backward chaining
PROLOG	Logic-based	Backward chaining
PTRANS	Rule-based	Forward chaining
PUFF	Rule-based	Backward chaining
RITA	Rule-based	Forward, backward chaining
S.1	Rule-based, Frame-based	Backward chaining
SAL	Rule-based	Forward chaining
SAVOIR	Rule-based	Backward and forward chaining
SPE	Rule-based	Forward chaining
SPERIL-I	Rule-based	Forward chaining
SPERIL-II	Rule-based	Forward and backward chaining
TALIB	Rule-based	Forward chaining
TATR	Rule-based	Forward chaining
TAXADVISOR	Rule-based	Backward chaining
THYROID MODEL	Rule-based	Forward chaining
WHEEZE	Frame-based	Backward and forward chaining
XCON	Rule-based	Forward chaining
XI PLUS	Rules, Induction	Control over direction
XSEL	Rule-based	Forward chaining
YES/MVS	Rule-based	Forward chaining

5.12. Object-oriented knowledge structure for Management of Child Fever

This work emphasizes the usefulness of O-O knowledge structure to improve consistency, understandability, maintainability, and modifiability of knowledge base. As an implementation tool, we have chosen object-oriented tool- Level5 Object. One can find the knowledge structure of 'Child Fever Management (CFM)', incorporated in the application 'a Web Accessible Knowledge-based Advisory System for Peadiatric Fever Management at home' in chapter 9.

5.13. Discussions

In this chapter, we have considered the vital issues of knowledge engineering. Here we have tried exploring the difficulties associated with knowledge acquisition. Potential sources used in this research have been pointed out. We have tried also to analyse the relative suitability of different KR schemes from the viewpoint of an expert system designer for the medical management domain. Our analysis finds object-oriented approach more suitable for the problem domain.

From the above analysis and consequent results it might be apparent that the object-oriented paradigm is a panacea for all the woes of knowledge engineering/abstraction/representation, but however, the paradigm does have some drawbacks [59]:

- Firstly, O-O approach has a long learning curve. The classical developers have to devote several months before they are skilled enough to start a project using O-O approach.

- As a relatively new technology, there might be unavailability of robust and reliable tools such as AI-languages or a shell. However, at present, there are some ES-shells using O-O technology (e.g. Level5 Object) are available in the market.

- The third problem may come from the very nature of abstraction. The reliability of the abstraction layer(s) should be sufficiently high so that there should not be any bug with these layer(s). These bugs are rarely trapped by the application layer due to the shielding property of abstraction. A careful design is, obviously, required to overcome this problem.

But, however, there is no doubt that O-O technology should certainly be needful in : (i) designing a complex system; (ii) developing a complex system; (iii) in maintaining the system; and (iv) in modifying the knowledge base of a system.

References

1. Edward A. Feigenbaum and Julian Feldman (editors). Computers and Thought, Robert E. Krieger Publishing, Malabar, Florida, 1981.
2. C. Lodbecke, P. Van Fenema, P. Powell. Co-opetition and Knowledge Transfer. The Database for Advances in Information System, vol.30, no. 2, pp.14–25, 1999.
3. S. Preston and C. Chapman. Knowledge Acquisition for Knowledge-based Engineering Systems, Int. J. Information Technology and Management, vol. 4, no.1, 2005.
4. J. Liebowitz. Knowledge Management: Learning from Knowledge Engineering, CRC Press, FL., 2001.
5. S. Stabb, R. Syuder, HP. Schnurr, Y. Sure. Knowledge processes and ontologies, IEEE Intelligent Systems 16, 1:26-34, 2001.
6. R. Weber, DW. Aha. Intelligent Delivery of Military Lessons learned, Decision Support Systems 34, 3:287-304, 2002.
7. David S. Prerau. Developing and Managing Expert Systems. Addison-Wesley Publishing Company, Inc., pp. 200, 1990.
8. K. L. McGraw and B. K. Harbison-Briggs. Knowledge Acquisition, Principles and Guidelines. Englewood cliffs, N. J. Prentice-Hall, 1989.
9. Efraim Turban. Expert Systems and Applied Artificial Intelligence. Macmillan Publishing Company; New York, 1992.
10. J. H. Boose. A survey of Knowledge Acquisition Techniques and Tools. Knowledge Acquisition, March 1989.
11. K. Bogel. Design and Implementations of a Web-based knowledge acquisition toolkit for medical expert consultation systems. Doctoral Thesis, Technical University of Vienna, 1997.
12. N. J. Nilsson. Principles of Artificial Intelligence. Narosa Publishing House, New Delhi, 1990.
13. Elaine Rich and Kevin Knight. Artificial Intelligence (2nd ed.). Tata McGraw-Hill, New Delhi. 1991.
14. Henry C. Mishkoff. Understanding Artificial Intelligence. H. W. Sams and Co. pp. 9-10, 1985.
15. Eugene Charniak and Drew McDermott. Introduction to Artificial Intelligence. Addison-Wesley. 1985.

16. K. S. Leung and M. H. Wong. An expert system shell using structured knowledge: An object-oriented approach. IEEE Computer; vol. 23, no.3. pp. 38-47, 1990.

17. D. W. Patterson. Introduction to Artificial Intelligence and Expert Systems. Prentice-Hall of India, New Delhi, pp. 47-79, 2000.

18. M Quillian. Semantic Memory. In Semantic Information Processing. Ed. M. Minsky. Cambridge: MIT Press. 1968.

19. John F. Sowa. Semantic Networks: Article in Encyclopedia of Artificial Intelligence, edited by Stuart C. Shapiro, Wiley, 1987, second edition, 1992.

20. M. Van et al. Emycin manual. Tech. report. Heuristic programming project. Stanford University. 1981.

21. F. R. Kehler. The role of frame-based representation in reasoning. Comm. ACM; vol.28, no.9. pp. 904-920, 1985.

22. D. A. Waterman. A Guide to expert systems. Addison-Wesley, Reading, Mass. 1986.

23. M. Minsky. A framework for representing knowledge. The Psychology of Computer Vision, (ed.), P. H. Winston. McGraw Hill, New York. 1975.

24. M. Minsky. A Framework for Representing Knowledge, MIT - AI Laboratory Memo 306, June, 1974. http://web.media.mit.edu/~minsky/papers. Accessed on 10/06/2003.

25. R. J. Brachman, H. J. Levesque. Knowledge representation and reasoning, Morgan Kaufmann Publishers, 2004.

26. J. F. Sowa. Knowledge representation. Logical, Philosophical and Computational Foundations, Brooks/Cole, 2000.

27. J. Graham, P. L. Jones. Expert System Knowledge Uncertainty and Decision, Chapman & Hall, 1988.

28. C. A. Welty. An integrated Representation for Software Development and Discovery, PhD thesis, Rensselaer Polytechnic Institute, 1996. http://www.cs.vassar.edu/faculty/welty/papers/phd/. Accessed on 10/10/2007.

29. J. S. Aikins. Prototypical Knowledge for Expert Systems: a Retrospective Analysis, Artificial Intelligence, Elsevier, pp.207-211, 1993.

30. M. Marinov. An Interactive tool for frame representation, Information Technologies & Control, No. 1, pp.16-19, 2003.

. Valkovska, J. Grundspenkis. Development of Frame Systems Shell for Learning of Knowledge Representation issues, Proceedings of conference Comp Sys Tech, pp.IV.11-1 - IV.11-6, 2005.

32. H. Alshawi, D. Carter, R. Crouch, S. Pulman, M. Rayner A. Smith. CLARE - A Contextual Reasoning and Cooperative Response Framework for the Core Languge Engine. Final report, SRI International, Cambridge UK, 1992. http://citeseer.ist.psu.edu/alshawi92clare.html. Accessed on 10/10/2007.

33. D. G. Babrow and T. Winograd. An Overview of KRL: a Knowledge Representation Language. Cognitive Science, 1(3). 1977.

34. I. P. Goldstein and R. B. Roberts. Nudge, a Knowledge Based Scheduling Program. In Proc. of the Fifth Int. Joint Conf. on Artificial Intelligence. pp.257-263, 1997.

35. R. J. Brachman. A Structural Paradigm for Representing Knowledge. Report No. 3605. Bolt Beranek and Newman Inc. Cambridge, Mass. 1978.

36. Roger C. Schank and G. Peter. Childers. The Cognitive Computer. Addison-Wesley. 1984.

37. A. Goldberg and D. Robson. Smalltalk-80: The language and its implementation. Addison-Wesley; NY. 1983.

38. R. S. Pressman. Software Engineering: A practitioner's approach. 3rd edn., McGraw-Hill; Inc. 1992.

39. Object-Oriented Requirements Analysis (course notebook). EVB Software Engineering. 1989.

40. S. L. Achour, M. Dojat, C. Rieux, P. Bierling, E. Lepage, J Am Med Inform Assoc 8, 351, July 01, 2001.

41. M.-J. Huang, M.-Y. Chen, Expert Systems with Applications 32, 658, 2007.

42. J. Fox, N. Johns, A. Rahmanzadeh, Artificial Intelligence in Medicine 14, 157, 1998.

43. P. Terenziani, G. Molino, M. Torchio, Artificial Intelligence in Medicine 23, 249, 2001.

44. L. Ohno-Machado et al., J Am Med Inform Assoc 5, 357, July 01, 1998.

45. M. A. Musen, S. W. Tu, A. K. Das, Y. Shahar, J Am Med Inform Assoc 3, 367, November 01, 1996.

46. G. Kong, Dong-Ling Xu and Jian-Bo Yang. Clinical Decision Support Systems: A review on knowledge representation and inference under uncertainties, International Journal of Computational Intelligence Systems, Vol.1, No. 2, 159-167, 2008.

47. J. Fox, N. Johns, A. Rahmanzadeh, Artificial Intelligence in Medicine 14, 157, 1998.

48. P. A. de Clercq, J. A. Blom, H. H. M. Korsten, A. Hasman, Artificial Intelligence in Medicine 31, 1, 2004.

49. P. Terenziani, G. Molino, M. Torchio, Artificial Intelligence in Medicine 23, 249, 2001.

50. S. Miksch, R. Kosara, Y. Shahar, P. Johnson, in Proceedings of the 4th International Conference on Artificial Intelligence Planning Systems 1998 (AAAI Press, Menlo Park, CA) pp.11-18, 1998.

51. K. Nygaard. Basic concepts in object-oriented programming. Sigplan Notices; vol.21, no. 10, pp. 117, 1986.

52. O. Dahl and K. Nygaard. Simula - A language for programming and description of discrete event systems. Introduction and Users Manual. Norwegian Computing Centre, Oslo, Norway. 1967.

53. J. Durkin. Expert Systems: Catalog of Applications. Intelligent Computer Systems; Inc., Akron, Ohio, 1993.

54. J. Durkin. Expert Systems: A view of the field. IEEE Expert. pp. 56- 63, April 1996.

55. W. B. Gevarter, The nature and evaluation of commercial expert system building tools. IEEE Computer; vol.20 No.5. pp. 24-41, 1987.

56. F. Hayes-Roth. Invited Talk. IEEE Compcon; San Francisco, CA. February 1987.

57. A. Synder. Encapsulation and inheritance in object-oriented programming languages. Proc. OOPSLA, ACM; New York. pp. 38-45, 1986.

58. James L. Alty. Expert System Building Tools. Topics in Expert System Design. G. Giuda and C. Tasso,(ed), Elsevier Science Publishers B. V., North Holand, pp. 193, 1989.

59. R. Gupta, W. H. Cheng, R. Gupta, I. Hardonag and M. A. Breuer. An object-oriented VLSI CAD Framework: A case study in rapid prototyping. IEEE Computer, pp.28-37, 1989.