Contents

Development of Design Techniques for Knowledge Systems

Nobuhiko Seike

< Keywords >

Knowledge Systems, System Design, Expert Systems, Knowledge Base, ISM

1. Introduction

In recent years there has been a great deal of activity in the area of developing knowledge systems, with the major aim being the creation of practical working systems. The term 'knowledge system' refers to computer systems which are constructed around the processing of knowledge in a non－procedural way. This is in opposition to mainstream computer systems which use procedural methods of computation. Expert systems are one example of knowledge systems. They are currently used for such applications as determining credit ratings in financial institutions, fault－detection in manufacturing systems etc., machine translation, and image/voice recognition.[1]

However, if we look at the design techniques used for developing these systems we can see that in fact a great number of them still employ techniques used in developing procedural systems. As a results we can see that as far as the design of knowledge based systems is concerned, there has not been the necessary shift away from the old techniques used in developing procedural systems.

Of course, knowledge systems vary a great deal in their design and configuration, depending on the intended objective of the system.However, no matter what the objective, it is still true to say that they are clearly defined as knowledge systems by the fact that they require a different type of development method to procedural systems, which operate on an algorithmic basis.

This paper will present a new kind of design technique for developing expert systems, founded on the use of knowledge bases. These techniques will be compared to current design techniques for developing procedural systems. At present, the most widely used design techniquesare[2] based on the sequential control method[3] which is characteristic of the Neumann – type computer. With these techniques the emphasis is put on creating algorithmic procedures, working from the initial system analysis stage right through to the final coding stage.

However, in developing systems using knowledge bases, such as expert systems, the emphasis shifts away from procedures on to the system development, where the main emphasis is the acquisition and representation of knowledge. Because of this, existing system development techniques are not fully able to cope with the need to acquire and represent knowledge.

Here, a new technique for system development and system design will be put forward . This is based on a predicate language which describes the logical structure of knowledge bases, and shows how to collect, organize and store information (facts and rules) which accumulates in the knowledge base.

The core features of this technique are

1) a method for knowledge acquisition based on a predicate language which expresses the structure of the knowledge base
2) management of knowledge using modularization
3) prototype modeling for aiding system development

2. The need for knowledge systems

It is now almost 50 years since computers came into use, and they are now used in every facet of daily life. The way they have been used is to proceduralize the actions and ideas of humans, store these procedures and perform tasks formerly carried out by humans. In virtually all of these cases the emphasis has been on the development of methods for programming and system design based on these procedures.

However, with the emergence of the so-called "information oriented society", which has been created through the improvement and enhancement of computer hardware and software, there has been an inevitable shift in data processing technology, away from procedural based programming towards programming technology based on knowledge systems.

The reason for this is that human behaviour arises from activities which use knowledge, such as association and creativity, and not simply as algorithmic behaviour.

As part of this shift, data processing technology is now able to perform a greater range of data processing functions. However the design and development of computer systems has not kept pace with these changes and is still heavily based on procedural methods of design.

This is the major reason why there has not been greater development of knowledge based systems, most of the work being left to only a handful of individual experts. As a result of this, the design and development of knowledge systems has used procedural based techniques with only a few superficial changes.

Even though there are still many open-ended questions on how to develop knowledge based systems, as this technology is in the formative

stage, there is a growing need for the establishment of effective system development and design methods for developing these type of systems.

Here, a workable specification for the system design and development of knowledge based systems will be presented by giving an example of the development of an expert system. The following is a proposal for a working specification for the design and development of a knowledge system, based on an actual example of an expert system.

3. Problems associated with the development of expert systems

Firstly, the problems associated with the development of an expert system will be discussed focusing on the configuration of the different functions in the system. The configuration of an expert system[4] is shown in Figure 1. The core modules of the configuration are the "knowledge base", "inference engine", and the "knowledge acquisition mechanism", "explanatory mechanism", "user interface mechanism" which make use of the knowledge base and inference engine.

When creating a new expert system it is generally the case that an existing inference engine, called an "expert shell", is used rather than creating a totally new inference engine. Because of this, in many cases, the knowledge base is created independently of the inference engine.

However, even though the system design is carried out without reference to the inference engine, if the system developer knows the inference methods used in the inference engine this will help with knowledge acquisition. Therefore, a method for aiding knowledge acquisition must be devised allowing the developer to incorporate information about the inference engine.

For example, when using a production system as a method of inference, because inferencing is carried out using IF – THEN relationships it is desirable to have a method for inputting information which can represent these relationships in a concrete way. Also, when using a frame system it is desirable to have a method for providing representations of the frame hierarchy structure and changes in results caused by parameter

variation.

Also, when using a production system it is desirable to have a chart which shows the inferencing processes. The reason for this is that the thought processes of the experts who provides the knowledge may vary. Also, the ability of the expert to correct knowledge when creating the knowledge system may also vary, depending on whether or not backward reasoning, forward reasoning or bidirectional reasoning are used.

However, because information about rules in the knowledge base is used in the building of the reasoning process, it is necessary to devise different methods for looking at rules to those used for deducing facts.

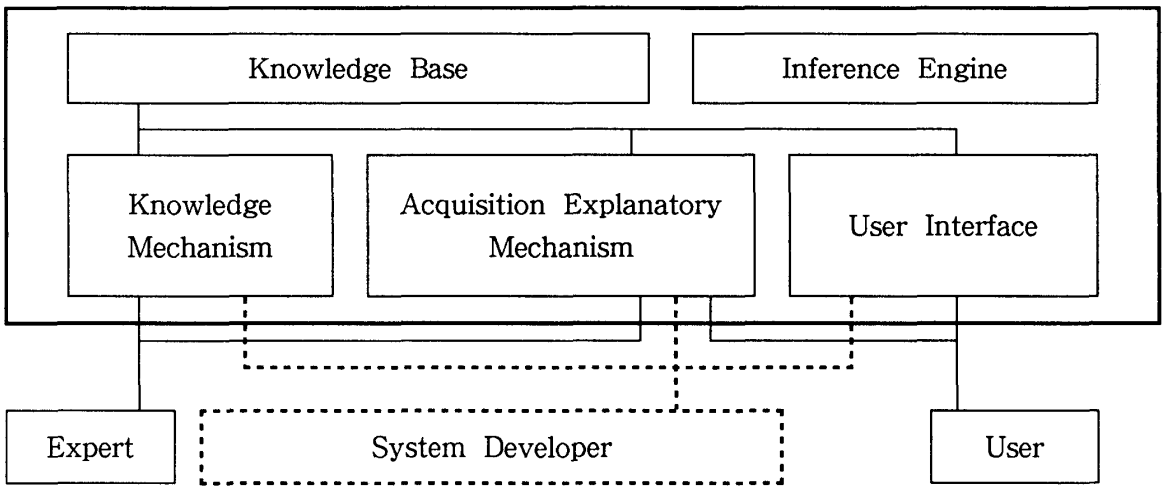

Figure 1 configulation of expert system

Next, we can summarize the problems associated with building an expert system, according to the individual roles of the participants in the project, in the following way. The two people who have the greatest influence on the process of building the system are the domain expert, who provides the knowledge of the domain, and the system developer who configures the system. The domain expert provides the knowledge used to create the knowledge base. However, this knowledge is of two types, conscious (actual) knowledge which is in an identifiable form, and unconscious (potential) knowledge of which even the expert himself may not be aware[5]. Therefore a special method of extracting the knowledge held by the expert

is required. There is also a need to classify the knowledge in terms of knowledge about facts, knowledge about procedures, and knowledge which shows relationships between facts.

On the other side of the process, the system developer configures the expert system using the knowledge given by the expert. As the system developer is not an expert in the domain knowledge it is necessary to devise a common base of communication between the expert and the developer in order to extract the domain knowledge from the expert.

The system developer will not have the level of understanding of the domain knowledge as the expert knowledge provider. Therefore it is essential to create a standard method for acquiring the information the expert wants to convey, and share information using this standard method of communication. At the same time, the system developer must look at the issues related to the user interface, so that the system is as easy to use as possible.

The user of the expert system does not have the level of knowledge about the domain as an expert, and will also be inexperienced in using an expert system on computer.

This means that functions such as explanations about the expert knowledge, help on using the system etc must be incorporated into the system.

It is possible to classify users of expert systems by the intended purpose of use. The expert himself may use the system to aid his own work. Non－experts will use the system acting as an expert. As a consequence of this, the knowledge in the system must be able to cope with various requests, such as offering the optimum solution for a problem, or providing information for showing alternative solutions, or evaluating alternative solutions etc. Consequently, the ways of presenting information will differ depending on the user.

We can thus see that there is a need for many more functions in an expert system than simply building a knowledge base and inference engine. Furthermore, it is also desirable to include the following functions: 1) a knowledge acquisition mechanism for efficiently acquiring knowledge

from the expert, 2) an explanatory mechanism for improving understanding of the ways the user and expertize use the system and 3) a user interface which makes the system easy to use.

We can summarize the stages of development of an expert system, based on the issues raised above.

Planning stage

1) estimation of the development timescale
   - estimate time for the creation of a prototype and correction
2) investigation of the aims of the user
   - consultation and provision of information
   - hypothesis of use by expert and non – expert
3) establishment of a common framework for liaison with the expert
   - focus on the expert as the source of knowledge

Preparation stage

1) selection of knowledge representation mode
   - decide on method of collecting information
2) method of acquiring knowledge
   - extraction of facts and relationships
   - method of investigating actual and potential knowledge

Selection satage of development tools

1) selection of computer equipment
   - selection of general – purpose or specialist computer
2) selection of computer language
   - trade – off between ease of development and flexibility
3) investigation of inference method
   - comparison of inference methods used by experts

Creation stage of development tools

1) creation of a specification for collecting knowledge
   - ease of collection of knowledge
2) creation of a specification for correcting knowledge
   - clarity of correction of knowledge
3) creation of a history log for correction of knowledge

Development stage

1) mechanism for acquiring knowledge
   - design of input and output
2) creation of prototype model
   - start of extended model
3) explanatory mechanism
   - investigation of facts and rules

Correction and enhancement stage

1) rule diagram
   - interrogation of facts and rules
2) modularization of rules
   - interrogation of facts and rules

Figure 2

Points to be taken into account at each level of development of an expert system

4. Comparison of the development of procedural systems and knowledge systems

In this section the development techniques for building an expert system which I have proposed will be compared with existing techniques for developing a procedural system. The stages in developing a procedural system are 1) system investigation 2) system analysis 3) system design (basic design/detailed design) 4) steps for system control[6].

(1) System investigation

System investigation involves carrying out an analysis of the target domain, determining what procedures are to be carried out (and in what order) and what problems are likely to arise. Design of a knowledge based system is the same at this stage.

However, design of a knowledge system differs in that the actual task to be carried out is not as clearly defined as the tasks which will be handled by a procedural system. The computerization of procedures which are in themself difficult to proceduralize is an unavoidable feature of knowledge based systems. Therefore, when developing a knowledge system, it is necessary to focus on how specialists solve problems without worrying about each individual procedure. Also, in developing a procedural system the emphasis is put on analysing the procedures of work which are currently employed and in making clear the flow of functions and information.

However, in developing a knowledge system the objective is not to create procedures but instead to describe and represent the actual knowledge of the expert. This means that the structure of the knowledge itself must be decide upon at this stage.

With a procedural system, for example a warehouse management system, it is relatively easy to decide on a structure for the procedures and functions of warehouse management, as the domain is limited and well defined. But with a knowledge system it is rare for boundaries to be fixed at the outset. Indeed the difference between the two types of system is that the knowledge system preserves the ability to expand the boundaries of the system.

(2) System analysis

With a procedural system a functional analysis is carried out and the relationships between functions and information groups are determined. With knowledge systems it is necessary to determine types of knowledge used in the system. It is possible to regard at the items which are stored in the knowledge base in the following way.

Real world facts are represented in the form of knowledge, meaning that as the real world becomes more complicated the form of representing it must also adapt to a more complicated form of representation. This entails that the structure of the knowledge base must also change to accommodate this increased complexity. However, if we look at interpreting the real world in the form of predicate logic expressions then we can describe it as follows[7]

1) define set D which is not empty
2) assign the value in D to the constants and variables contained in logical expressions
3) the functions from Dn to D are assigned by the function symbols of each n variable
4) the relationship defined in Dn by the function symbols of each n variable predicate are assigned
5) true or false values are assigned in the logical expressions

With expert systems, the collection and storage of facts and the inferencing which is based on rules uses this concept. Therefore verification is carried out on the hypothesis to be verified according on truth/false conditions, by creating a set of non－empty facts. The facts to be stored in the expert system are collected once the framework for storing the facts has been determined. However, it is important to start with a totally open－ended framework as outlined above.

(3) System design

With either type of system this stage involves designing input and output processes. However, with a knowledge system a lot of thought must be put into the help system and user interface. There is also a major difference

in the design of files and processes.

In designing input processes for knowledge systems as the knowledge is provided by an expert, it is necessary to design for the input of facts, rules, and certainty. It is also necessary to design input procedures which allow different forms of interrogation by the user.

With output processes, it is necessary to design output for displaying inference results to the expert and user, separate display of rule tables for checking by the expert, and separate help displays for the user.

With file design, the files in a procedural system are dependent on the program, but with a knowledge system the knowledge base is designed independently of the inference mechanism. Therefore file design is not necessary with an expert system, except for the design of the physical storage structure and the structure of the knowledge base itself. Put another way, although file design in necessary it is not essential when developing an expert system which simply adds knowledge to an existing knowledge base.

Furthermore, with developing knowledge systems, if there is no development of an inference mechanism then there is no process design stage.

(4) System control

There is no great difference in the computer processing involved at the system control stage. The same cycle of programming > testing > debugging until a completed system is reached is involved. However, with a knowledge system the difference is that the 'completed' system is really no more than an intermediate system. Therefore it is necessary to have a knowledge acquisition mechanism which allows for the correction of knowledge, in place of the error lists, trace lists and dump prints which are used in procedural systems.

Figure 3 shows the flow of development for each system

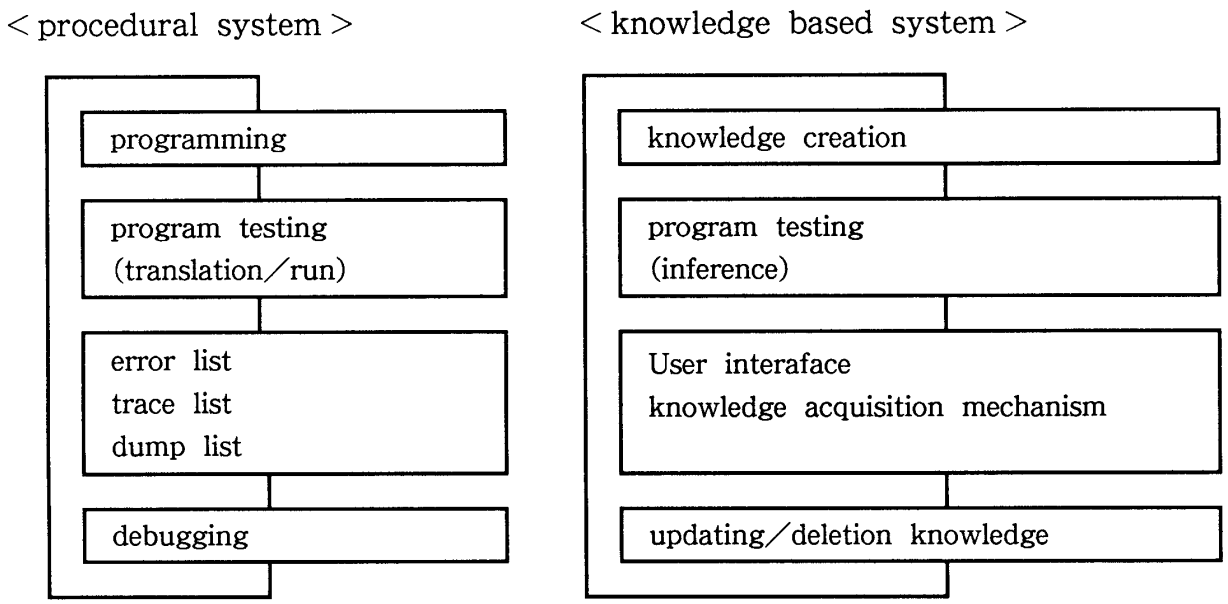

Figure 3 Comparison of system development

Table 1 gives the items to be implemented at each stage for both types of systems

Table 1

Comparison of system development of procedural v knowledge systems

	development systems	knowledge systems
Functional Analysis	Analysis of Domain Comparison of Different System	Determine Problems Comparison of Procedural System
Function Analysis	Chart of Function – Information	Hearing
System Design	Design of Input Design of Output File Design Process Design	Design of Fact Table Rule Relationship Table Certainty Decision Chart Goal Chart
System Creation	Program Creation	Knowledge Base Creation (Inference Engine Creation)

5. Capturing of knowledge for designing a knowledge system

As described above, the major characteristic of an expert system is that the programming and data are quite separate, unlike traditional programming in a procedural system. Because the inference engine (the program) exists independently of the knowledge base (the data), the development of a knowledge base is normally carried out independently from the development of the inference engine

Let us now look at the thinking behind the development of a knowledge base. The knowledge held in the knowledge base of a production system contains knowledge used as facts and knowledge used as rules which express the relationships between the facts[8]. This is the same relationship, between meaning and form, as can be found in the rules used in natural language. If we look at this fact in terms of predicate language then we need to establish several points from the outset.

The first is the definition of terms. Terms are constants, variables and functions. Constants are concepts which refer to specific objects, for example "Tokyo". Variables are concepts which express arbitrary areas in which specified objects can be placed, e.g. "City". Functions express the relationships between variables and constants, for example the function "I am a teacher". Functions are expressions in which defined terms are linked together using function symbols to show the relationship between the terms. Predicates are expressions in which fixed terms are linked together using predicate symbols to show the declarations for each term.

Predicate logic expressions can be defined by the following three conditions :

1) predicates are predicate logic expressions
2) if we take P, Q as predicate logic expressions then

   $\overline{P}$, $P \wedge Q$, $P \vee Q$, $P \rightarrow Q$, $P \rightleftarrows Q$ are predicate logic expressions the symbols mean

   $\overline{\ }$ is NOT, $\wedge$ is AND, $\vee$ is OR, $\rightarrow$ is INCLUDES, $\rightleftarrows$ is IF and ONLY IF
3) if we take P (X) as a predicate logical expression then

$(\forall x)P(X), (\exists X)P(X)$ is a logical expression.

These type of expressions occur frequently when the system developer carries out "hearing" from the expert and are representative of the core elements of the knowledge system. For example, if we take P to mean "Japanese work hard" and Q to mean "Today is Saturday" then $\overline{P}$ means "Japanese do not work hard", $P \wedge Q$ means "Japanese work hard and today is Saturday", $P \vee Q$ is "Japanese work hard or today is Saturday", $P \rightarrow Q$ is "If Japanese work hard then today is Saturday", $P \rightleftarrows Q$ is "It is Saturday only when Japanese work hard and it can only be Saturday when Japanese work hard".

Also, it is possible to perform a number of transformations on logical expressions so that the same concept can be expressed in a different way. For example $P \rightarrow Q$ can be expressed as $\overline{P} \wedge Q$. If P then Q expresses the same meaning as if not P or Q.

However it is not the case that the transfer of knowledge is carried out exclusively in this form, during hearing sessions between the expert and the system developer. The everyday natural language which humans use to communicate is far more flexible than this, meaning that the system developer must convert natural language into predicate language.

Moreover, with natural conversation we interpret the meaning of utterances using background knowledge, meaning that the knowledge in an expert system is insufficient if it is based solely on what is said in conversations between the expert and the developer. Therefore we must convert the knowledge conveyed by natural language and not just the utterances, into a fixed form using predicate language.

If we can perform this task using a strictly defined structure then we can capture knowledge in a much better form as predicate language. In order to achieve this, we need some form of specification, even if it is in a rudimentary form, for carrying out these dialogues .

One problem which arises when building a knowledge base, is that

there are often cases where we need to think about the different ways of expressing knowledge. For example, in the case of the word "Tokyo" it is not always possible to decide on whether or not this is a constant or variable simply by looking at the word "Tokyo" itself. If the word Tokyo cannot be further split into more constituents then it will be processed as a constant. However, we can see that the word "Tokyo" also consists of Itabashi Ward, Musashino City and so on meaning that it could be treated as a variable. It is up to the system developer to decide on whether to treat such words as constants or variables when he is carrying out "hearing" sessions with the expert.

A different kind of problem arises when referring to concepts such as "consumer". Here the decision has to be made as to whether it refers to all consumers (∀) or to an individual consumer (∃). If it is decided that it refers to all consumers then we can use it is a rule. If it refers to all consumers then it can be used as a rule, but if it refers to a specific consumer then it must be treated as knowledge about a fact and other conditions have to be added.

Apart from these examples, there is the general problem associated with natural language whereby there are utterances which are not true, the subject or predicate are not clear and so on.

This can introduce complications in the knowledge building process, in that 1) the uncertainty can be due to the expert expressing the concept as an uncertain concept, or 2) it has not been conveyed clearly to the system developer, or 3) the system developer himself is treating the concept in an unclear way.

6. System development and design of expert systems

I will now set out in more detail the separate stages needed for developing a knowledge system in order to resolve the problems mentioned so far. Figure 4 shows the development steps for building a knowledge system. The stages which are the same for a procedural system have been omitted.

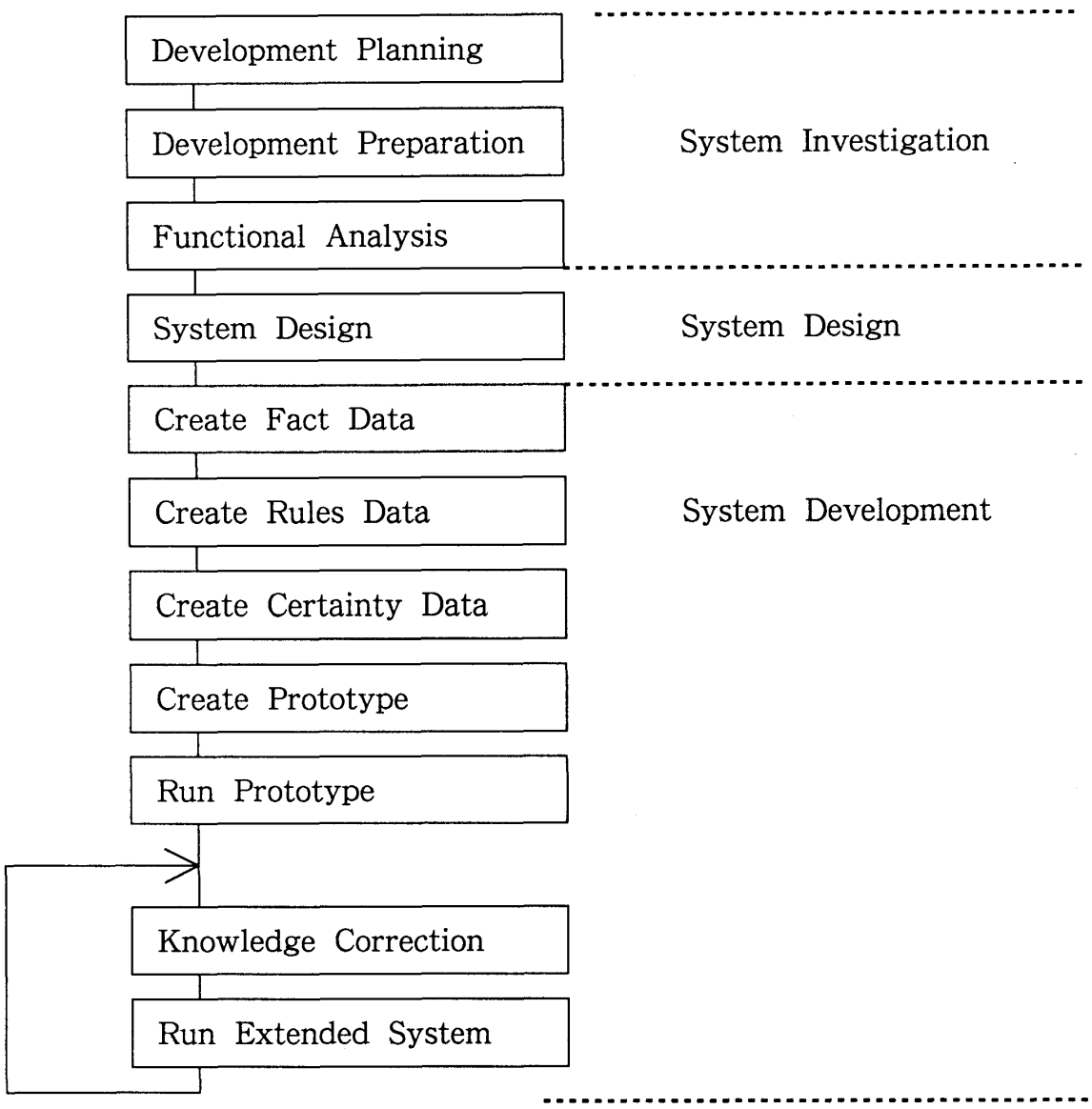

Figure 4 Development steps or a knowledge system

6.1 Development planning and preparation

The first thing to consider at the planning stage is the issue of whether or not the domain being looked at should be computerized at all, and if the computerization can be handled by a procedural system. When investigating whether or not to start the project, all of the problems associated with the target domain should be made clear from the outset. It is then normal procedure to decide on whether or not to carry out the work using manual methods or computerization by looking at factors such as costs, efficiency gains and ease of use. In this sense, the decisions

to be made at the planning stage are the same as for developing a procedural system.

However, with a knowledge system it is extremely important to realize that the final form of the system will still be unclear at this stage. It is important to recognize that the necessity for computerizing the target domain will be decided by the requirements of the user. Needless to say, there are many types of expert system. One way of evaluating computer systems is by how much time has been saved in performing a given task. But with an expert system it is often the case that the evaluation of an expert system will also take into account the demands of the user as an expert system is used to aid the intellectual task of an expert.

When it has been decided to computerize the target domain the next stage is to decide on whether to use a procedural system or knowledge system. The kind of problems for which knowledge systems are suited, and for which a procedural system will have difficulties, are as follows:

1) problems for which the optimum solution cannot be determined
2) problems where the optimum solution is not required
3) problems where a large number of conditions must be included in the processing making the procedure very complex

The decision to start developing a knowledge system should only be taken after it has been decided that a procedural system cannot handle the requirements of the target domain. If the decision to develop a knowledge system is taken, then a critical condition for using a knowledge system is that it can cope with the problems which the procedural system cannot.

The next factor to take into account is the feasibility of whether or not the knowledge of the expert can be structuralized. Knowledge which is well defined and structured, such as legal or contractual language is relatively easy to collect as actual knowledge, whereas knowledge which is of a heavily creative nature, such as language used in design etc. is hard to collect.

Finally, it is necessary to estimate the timescale and the labour/budgetary costs. With an expert system it is essential to split this estimation into the time for developing a prototype and the time for developing a completed system. The reason for this is that the development of the final system starts after the prototype model is finished.

6.2 Functional analysis

The main problem which occurs in the creation of an expert system is how to acquire the domain knowledge from the expert. Many of the expert systems we can see today are open to the criticism that they are not really expert systems at all, because the domain knowledge of the expert has not been fully included in the system. Therefore it is essential to devote a sufficient amount of time to the issue of acquiring knowledge from the expert.

The following points should be taken into account when planning how to carry out the knowledge acquisition process :

1) what form should experiential knowledge take
2) how should potential knowledge be turned into actual knowledge
3) how should the knowledge be summarized
4) how should certainty be defined

The "hearing" stage begins with the system developer asking the expert about the way the task is carried out, the different stages involved, and the information used when carrying out the task. As there are usually many variations in the ways a task can be carried out, it is generally believed that there is no common specification chart which can be used for the hearing process.

However, even though this is an intellectual activity, it still consists of goals, information to support the task and some form of systematization, meaning that we can create such a specification chart based on common criteria. These criteria are the framework, problem solving method, supporting information, final results of the task. These are shown in Figure 5.

Basic Hearing Specific Chart		Date	
Specific Chart No.		System name	
Knowledge Provider		Developer	
Outline of Specialist Tasks			
Case Study of Task			
Task Items			
Problems and Methods for Solution			
Related Information			
Results Expected by User			

Figure 5 Basic Hearing Specification Chart

At the start of the hearing stage it is important not to ask for detailed facts and solutions. Rather, it is important to make clear the range of knowledge held by the expert by obtaining the different types of conscious knowledge, keywords on potential knowledge, opinions on certainty etc.

Detailed Hearing Specific Chart		Date	
Specific Chart No.		System name	
Knowledge Provider		Developer	
Case No.		Case Name	
Case Outline			
Unique Features of the Case			
Solutions			

Figure 6 Detailed Hearing Specification Chart

When gathering knowledge from the expert it is good practice to list up several typical examples of the task and solve each one by looking at the methods used.

Fact Enquiry Chart			Date	
Specific Chart No.			System name	
Knowledge Provider			Developer	
Case No.			Case Name	

Fact Table

Fact No.	Entity No.	Entity	Predicate No.	State or Action

< Method of Entering Data >
Fact No. : sequential numbers to distinguish facts
Entity No. : number to distinguish entities
Entity : main constituent of the state or action
Predicate No. : number to distinguish state or action
State or Action : explanation of the main constituent

Figure 7 Fact Enquiry Chart

The Fact Enquiry Chart is used for transforming facts, states and actions into predicate forms. It can also be used with a Rule Enquiry Chart to check whether or not there are facts which are not used by the rules, or if there are missing descriptions for facts which are used in rules etc.

The enquiry starts from actual knowledge about the specialist tasks but may then move on to potential knowledge held by the expert as a way of integrating the two types of data together.

Rule Enquiry Chart		Date	
Specific Chart No.		System name	
Knowledge Provider		Developer	

Number	Condition Fact	no.	Conclusion Fact	no.

Number : rule number
Condition : if...
Conclusion : ... then
Fact No. : fact number from the fact enquiry chart.

Figure 8 Rule Enquiry Chart

The Rule Enquiry Chart is used for listing the rules which link the facts collected in the knowledge base, and takes expressions in a single sentence form. However, as this is later converted into the form used by the Rule Specification Chart for the input of knowledge, it is necessary to make clear any multiple conditions in the condition - part of rules in the Rule Enquiry Chart.

6.3 System design for a production system

As described above, there are many ways of representing knowledge in an expert system. The focus in this section will be on a method of design for a knowledge system which uses a production system as the method of inference.

< Method of Designing Facts >

The Fact Design Chart is used for aiding the storage of facts. At this stage the following are points should be kept in mind :

1) how should experiential knowledge which has been acquired using the Fact Enquiry Chart be formalised
2) how the potential knowledge left over is to be actual
3) how to summarize the knowledge

As this Fact Design Chart will be used for the correction, deletion and addition of facts after the prototype system has been created, it is necessary to create this chart in a form which can be evaluated as information for discriminating between individual facts.

Fact Design Chart		Date	
Specific Chart No.		System name	
Knowledge Provider		Developer	

Fact Table

Fact No.	Content	Necessity
		5 4 3 2 1
		5 4 3 2 1
		5 4 3 2 1
		5 4 3 2 1
		5 4 3 2 1
		5 4 3 2 1
		5 4 3 2 1

Fact No. : continuous numbers to discriminate between facts
Content : includes the fact and state/action
Necessity : 5 level evaluation
5 = 100 % necessary fact
3 = standard fact which is normally used
1 = facts which are unlikely to be used

Figure 9 Fact Design Chart

< Method for Rule Design >

As with the Fact Design Chart, this method is used for systematizing the rules decided on by the rule enquiry process. In particular, as it uses a form of representation using nodes and arrows to show how the rules are related it can also be used for aiding the correction and addition of rules once the prototype system has been built. When interrogating the rules there is a need to discriminate between conditions which can be separated from other conditions, and those which cannot.

It is also necessary to define the certainty of each rule. The certainty is a measurement of how to treat the level of certainty of that rule.

This specification chart can add a level of necessity as an aid for interrogating the rule system. Certainty is one method of knowledge representation, whereas level of necessity is a tool for analysing rules.

Rule Design Chart				Date			
Specific Chart No.				System name			
Knowledge Provider				Developer			
Fact		Higher Order Fact		Parallel Fact		Lower Order Fact	
No.	Content	No.	Content	No.	Content	No.	Content
1							
2							
3							
4							

Number : fact number
Higher Order Fact : fact which is a condition of the relevant fact
Parallel Fact : fact which is conjoined by AND or OR with the relevant fact
Lower Order Fact : fact which is introduced by the relevant fact

Figure 10 Rule Design Chart

< Method for deciding on the Certainty >

As yet there is no generalized method for specifying the certainty of rules in an expert system.

Here I propose a method of interrogating certainty using the concept of ranking rules by certainty, together with the Delphi method.When specifying the certainty for each rule, the value is no more than a subjective value

assigned by the individual expert. Because of this, there can be no objective comparison with other rules or between different experts.

First of all, an ascending sort is performed on the condition – part of the rules for rules with the same conditions. Then it is possible to decide on the certainty according to that ranking.

This technique can also be used on rules which have the same condition – part. Next, an effective method for correcting certainty is to compare the decisions made by different experts. However as there is variation in the decisions made by each expert, it is not possible to rigidly measure certainty and so correction using Delphi method must be employed.

Figure 11 shows the chart for deciding on the certainty taking these factors into account

Certainty Design Chart		Date	
Specific Chart No.		System name	
Knowledge Provider		Developer	

Certainty Table

Rule No.	Content	Certainty

Rule No. : rule number
Content : description of rule
Certainty : +1 = 100 % positive
−1 = 100 % negative
0 = no certainty

Figure 11 Certainty Decision Chart

< Design of knowledge acquisition mechanism >

Firstly, it is necessary to design the initial input for gathering knowledge from the expert and for correcting this knowledge once the prototype is completed.

It is also desirable to create a method for displaying rule tables which can be use to aid knowledge correction.

< Design of help system for the expert >

The help system for the expert should provide the following assistance:

1) explanation of the inferencing used to reach the final goal
2) display the next best inference results according to certainty

The first explanation will show how the conclusion (goal) was reached and by what inference path. It is also used for making sure the inference path is the same as the one used by the expert. The second explanation is used for investigating the possibility of discovering better paths to the goal, by correcting the certainty which was initially defined.

< Design of help system for the user >

The help system for the user should provide the following assistance:

1) explanation of basic vocabulary used in the system
2) a manual for using the system
3) explanation of reasons for displaying problems
4) explanation of inferences used in the rule conclusions

The first two help functions provide the user with supplementary knowledge about the domain handled by the expert system and the structure of the expert system itself.

The third help function provides explanations of the terminology which appears in questions, and the aims of the questions which are presented. The fourth help function provides explanations on the reasoning carried out, although is not as detailed as the reasoning process of the expert. The main function is to provide explanations on rules which are used directly in deciding the final outcome of the goals and explaining why those particular rules were used.

< Design of the user interface >

In the development of a expert system, it is essential to invest a lot of effort on the user interface. There are many areas in which user interfaces can be improved, and many ways of doing so, but it is especially important to pay a lot of attention to the processes involved in correcting the knowledge provided by the expert.

The reason for this is to make sure the thought processes of the expert are not interrupted whilst knowledge correction is being carried out.

The operation of an expert system should be as smooth as possible, from analyzing the knowledge map, to making clear any problems which arise and building it as new knowledge.

In order to do this, links between windows on the screen should be seamless and it should be easy to carry out parallel process. This kind of interface requires an environment which supports multi – windowing and multi – tasking.

6.4 Method of knowledge interrogation

The final issue in developing an expert system relates to program debugging. The kind of program debugging which is carried out in a procedural system cannot be carried out in the same way with an expert system. The correction of logic errors in a procedural system, take the form of correction of logic in the development of an expert system, this correction occurring mid - way through the development process.

We can summarize the main issues related to logic correction in a expert system as follows :

1) in what form should correction and addition of facts be carried out ?
2) how should the logic flow be represented
3) how should the correction and addition of rules be carried out
4) how should the correction of certainty be carried out
5) how should the knowledge of the expert be extracted

Based on these five points, there are two methods of logic correction which we can employ, the first using the Fact Relation Chart and the second using Modularization.

Fact Relationship Chart

When correcting knowledge related to the facts acquired from the expert to see if it is valid, the validity of the facts themselves are not under scrutiny, rather the relationship between the facts and the rules are what is looked at.

Therefore it is necessary to create a chart of the representations (rules) of how the facts are linked together. For any given fact it is necessary to make clear what items it contains as the condition - part (IF items/ superordinate items). Also, if the conditional - part is conditional on multiple facts it is necessary to make clear what kind of multiple condition it is. Furthermore if the condition itself is a complex condition it is necessary to make that clear.

Figure 12 is a Fact Relationship Chart based on these points

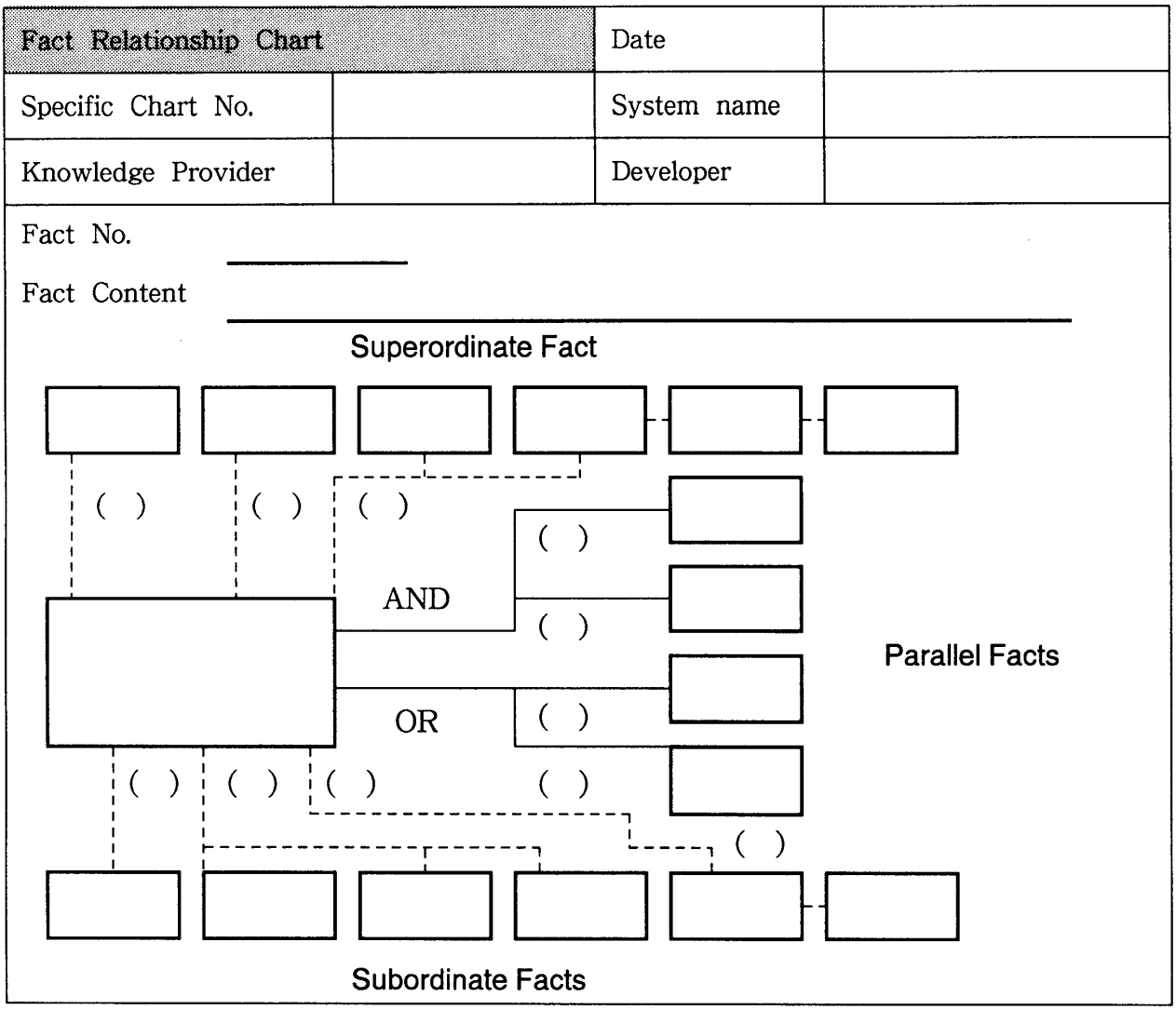

Figure 12 Fact Relationship Chart

Modularization

With modularization it is important that the information gained as a result of the modularization is useful for rediscovering the knowledge of the expert.

For example, in a case such as the one shown in Figure 13 where the relationship between facts is represented in result arrays, it is possible to classify the elements a b c and d e f as different groups and make them into modules. However, with an inferencing method where the level of relationships between facts is evaluated as the certainty, even if the

relationship is tentative, there is the possibility that the facts will be linked by the presence of a level of relationship between those facts.

$$
\begin{array}{l}
\\
<\text{Reason}>
\end{array}
\begin{array}{c}
<\text{Result}> \\
\begin{array}{c} \\ a \\ b \\ c \\ d \\ e \\ f \end{array}
\begin{array}{c} abcdef \\ \left(\begin{array}{c} 111000 \\ 111000 \\ 111000 \\ 000111 \\ 000111 \\ 000111 \end{array}\right) \end{array}
\end{array}
\Rightarrow
\begin{array}{cc}
abc & def \\
\left(\begin{array}{c} 111 \\ 111 \\ 111 \end{array}\right) & \left(\begin{array}{c} 111 \\ 111 \\ 111 \end{array}\right) \\
\begin{array}{c} a \\ b \\ c \end{array} & \begin{array}{c} d \\ e \\ f \end{array}
\end{array}
$$

Figure 13 Example of possibility for modularization

With modularization it is necessary to build in a method which makes it possible to arbitrarily cut the certainty at a certain level. In the example in Figure 13 it is possible to modularize the relationship between the rules even further if we use only those items which have a certainty of over 0.5.

$$
\begin{array}{l}
\\
<\text{Reason}>
\end{array}
\begin{array}{c}
<\text{Result}> \\
\begin{array}{c} \\ a \\ b \\ c \\ d \\ e \\ f \end{array}
\begin{array}{c} a \quad b \quad c \quad d \quad e \quad f \\ \left(\begin{array}{cccccc} 0.8 & 0.6 & 0.9 & 0.3 & 0.4 & 0.5 \\ 0.7 & 0.9 & 1.0 & 0.4 & 0.2 & 0.0 \\ 0.6 & 0.6 & 0.6 & 0.3 & 0.2 & 0.4 \\ 0.3 & 0.2 & 0.4 & 1.0 & 0.8 & 0.9 \\ 0.4 & 0.4 & 0.2 & 0.8 & 0.9 & 0.5 \\ 0.3 & 0.4 & 0.1 & 0.7 & 0.5 & 0.9 \end{array}\right) \end{array}
\end{array}
\Rightarrow
\begin{array}{l}
\\
<\text{Reason}>
\end{array}
\begin{array}{c}
<\text{Result}> \\
\begin{array}{c} \\ a \\ b \\ c \\ d \\ e \\ f \end{array}
\begin{array}{c} a\ b\ c\ d\ e\ f \\ \left(\begin{array}{cccccc} 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{array}\right) \end{array}
\end{array}
$$

Figure 14 example of possibility for pseudo modularization (1)

If we leave out a specific fact, as in figure 15, it is necessary to build in a facility for discovering facts which can be further broken down into smaller modules, and provide a method for expressing the structure of each module if we leave out that fact.

$$
\begin{array}{l}
\\
<\text{Reason}>
\end{array}
\begin{array}{c}
<\text{Result}> \\
\begin{array}{c} \\ a \\ b \\ c \\ d \\ e \\ f \end{array}
\begin{array}{c} abcdef \\ \left(\begin{array}{c} 111000 \\ 111000 \\ 111100 \\ 001111 \\ 000111 \\ 000111 \end{array}\right. \end{array}
\end{array}
\quad \Rightarrow \quad
\begin{array}{l}
\\
<\text{Reason}>
\end{array}
\begin{array}{c}
<\text{Result}> \\
\begin{array}{c} \\ a \\ b \\ c \\ d \\ e \\ f \end{array}
\begin{array}{c} abcdef \\ \left(\begin{array}{c} 111000 \\ 111000 \\ 111000 \\ 000111 \\ 000111 \\ 000111 \end{array}\right. \end{array}
\end{array}
$$

Figure 15 example of possibility for pseudo modularization (2)

If we use these results, we can systematize the relationship between rules and facts using a structuralization technique employing ISM (Interpretive Structural Modelling). These are effective methods for the formation of expert knowledge.

7 ISM procedure

ISM is a technique[9] proposed by Warfield of the American Battel Research institute, which is used for structuralizing the elements which make up a knowledge system. It is a powerful method for assisting the structuralization of rules as will be shown below.

< ISM procedure >

1. extract the elements to be analysed
2. Perform relationship (in one direction) for each element
   1 = there is a relationship 0 = there is no relationship
3. Represent the relationships as an array
4. Add 1 to the opposite element in the array < treat it as array A)
5. Ask for the product of the array < A x A > < take the value of the product as A^2 >
6. End if A^2 is equal to A, if not then calculate A^2 x A. < A^3 > Repeat this procedure until $A^n = A^{n-1}$.
7. Call the resulting matrix the reachable matrix
8. Extract the descendant element or ascendant element for each element from the reachable matrix
9. Create a common set of the ascendant elements and descendant element for each element

10. structuralize
    1. In each element treat the ascendant elements and common elements equally as a rank
    2. Repeat the process deleting the parts which have been decided upon as ranks

Figure 16 shows a family tree which will be used to explain ISM, with figure 17 representing a family tree as an array, and figure 18 showing a reachable matrix

Example) Family Tree

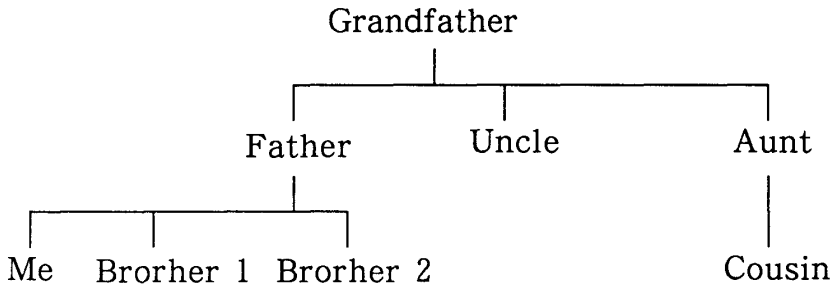

Figure 16 Family Tree

		Grand father	Father	Uncle	Aunt	Me	Brother 1	Brother 2	Cousin
	Grandfather	0	1	1	1	0	0	0	0
	Father	0	0	0	0	1	1	1	0
	Uncle	0	0	0	0	0	0	0	0
A^1	Aunt	0	0	0	0	0	0	0	1
	Me	0	0	0	0	0	0	0	0
	Brother 1	0	0	0	0	0	0	0	0
	Brother 2	0	0	0	0	0	0	0	0
	Cousin	0	0	0	0	0	0	0	0

Figure 17 Array Expression of a Family Tree

		Grand father	Father	Uncle	Aunt	Me	Brother 1	Brother 2	Cousin
	Grandfather	1	1	1	1	1	1	1	1
	Father	0	1	0	0	1	1	1	0
	Uncle	0	0	1	0	0	0	0	0
A^2	Aunt	0	0	0	1	0	0	0	1
	Me	0	0	0	0	1	0	0	0
	Brother 1	0	0	0	0	0	1	0	0
	Brother 2	0	0	0	0	0	0	1	0
	Cousin	0	0	0	0	0	0	0	1

Figure 18 Reachable Matrix of a Family Tree

Figures 19(1), 19(2), 19(3) are products of the reachable matrix shown in Figure 18

Element	Descendant Element	Ascendant Element	Common Element
Grandfather	1	1・2・3・4・5・6・7・8	1
Father	1・2	2・5・6・7	2
Uncle	1・3	3	3
Aunt	1・4	4・8	4
Me	1・2・5	5	5
Blother 1	1・2・6	6	6
Blother 2	1・2・7	7	7
Cousin	1・4・8	8	8

Figure 19(1) Elements of Reachable Matrix – Step 1

Element	Descendant Element	Ascendant Element	Common Element
Father	2	2・5・6・7	2
Uncle	3	3	3
Aunt	4	4・8	4
Me	2・5	5	5
Blother 1	2・6	6	6
Blother 2	2・7	7	7
Cousin	4・8	8	8

Figure 19(2) Elements of Reachable Matrix – Step 2

Element	Descendant Element	Ascendant Element	Common Element
Me	5	5	5
Blother 1	6	6	6
Blother 2	7	7	7
Cousin	8	8	8

Figure 19(3) Elements of Reachable Matrix – Step 3

By using these steps it is possible to divide the elements into ranks as shown in figure 20. It is also possible to recreate the first family tree from the indirect relationships between the elements.

< Result >

First rank 1

Second rank 2 3 4

Third rank 5 6 7 8

Figure 20 Ranking of Elements

By using these methods it is possible to recreate the whole family tree from the indirect relationships between superordinate and subordinate elements, using the Fact Relation Chart in Figure 16 even without the original family tree.

7. Method for correcting knowledge

Here a specification for a knowledge system based on this development procedure will be put forward . The use of a correction specification such as the one which will be outlined below will have a significant effect on the quality of the finished expert system.

< Correction of Facts >

Leaving aside corrections of careless mistakes in word usage, the correction of facts after the prototype system is finished involves altering the definitions of facts, actions and states. The following kinds of correction normally take place :

1) cases where the same object can be represented by a different

expression

2) case where inclusive concepts and the concepts included I them are used together

3) cases where there are facts not used by any rules

In any of these cases it is possible to find mistakes by using a truth relation chart.

When building an expert system it is extremely important to keep a log of corrections in the specification chart. It is also important to list in detail the corrections and why they occurred and not simply just the result of the correction.

Specification for Adding Corrections to Facts				Date	
Specific Chart No.			System name		
Knowledge Provider			Developer		
Table of fact corrections & additions					
Fact No.	Fact Content		No	Reason for Correction	Altered segment
	Entity				
	Predicate				
	Entity				
	Predicate				
	Entity				
	Predicate				
	Entity				
	Predicate				
Reason : reason for addition/change Altered segment : new, altered, deleted segment					

Figure 21 Specification for adding corrections to facts

< Correction of rules >

As with the Fact Relation Chart, it is extremely useful to represent the structure of rules using graphical representations.

Rule Structure Chart			Date	
Specific Chart No.		System name		
Knowledge Provider		Developer		

Rule No. _____

Rule Content _____

Superordinate rules

Conditional section	Conditional section	Conditional section
Certainty ( )	( )	( )

Current rule

Certainty ( )	( )	( )
Conditional section	Conditional section	· Conditional section

Subordinate rules

Conditional section : Superordinate rules condition – part
Current rule : condition – part and conclusion – part
conclusion section : conclusion – part of the subordinate rules
Certainty : certainty of rules

Figure 22 Rule Structure Chart

By using the Fact Relationship Chart we can systematize how a given fact is related to other facts using the rule descriptions as a guide. With the Rule Relationship Chart we can show the the flow of reasoning, i.e. the relationships between rules, and analyze the system using larger units of analysis. This can also be used for exploring whether or not links between the rules, as expressed in the inferencing processes in the

production system, represent the way of thinking of the expert. Displaying groups of rules on three levels, can be very useful for analysing the inferencing processes and subconscious knowledge of the expert.

Rule correction addition specification				Date	
Specific Chart No.			System name		
Knowledge Provider			Developer		
Table of rule correction & additions					
Rule no.	Rule Content		no.	Reason for Correction	Altered segment
	Conditional section				
	Conclusion section				
	Conditional section				
	Conclusion section				
	Conditional section				
	Conclusion section				
	Conditional section				
	Conclusion section				
Reason for Correction : reason for addition/change Altered segment : new, altered, deleted segment					

Figure 23 Rule correction addition specification

In the rule correction addition specification it is necessary to describe in detail the reason for the correction.

< Correction of Certainty >

The certainty used in an expert system is decided on subjectively by the domain expert and is therefore not validated by looking at each rule. Therefore the stability of the certainty can be considered to be low. As a result it generally arises that the initial settings designated by the expert have to be adjusted.

When deciding on what to base the adjustments of the certainty on, it is often the case that it is determined by the validity of results gained from imposed conditions.

An expert system is not always use to discover the optimum solution for a set of circumstances. On the contrary, there may be alternative solutions which the expert regards as valid, and the conditions which are used to bring out that solution are the knowledge of the expert. Thus it is sufficient to be able to display information which will show how to alter the certainty so as to arrive at the target solution.

In order to do this, one method is to enhance the decision making process of the expert by displaying a large number of results gained from simulations where the certainty has been adjusted. An effective system will display the results for each case where the settings for the certainty is changed in fixed stages from 0 to 1.

One problem faced when using this method is deciding which rules to include in the simulation. It is not possible to answer this question fully here but one method is to start the simulation using rules with the highest relevance factor as described above. Again, it is important to keep a log of the corrections made in this process, and to keep a detailed record of the reasons why the corrections were made.

Specification for Correcting Certainty			Date	
Specific Chart No.		System name		
Knowledge Provider		Developer		

Table of Certainty

Rule No.	rule Contents	Reason for Alteration	certainty
			→
			→
			→
			→
			→
			→
			→

Rule No. : rule number
Rule Contents : description of the rule
Certainty : +1 when 100 % positive, −1 when 100 % negative
0 when no certainty
when altered put figure before and after the →
when a new certainty is created put x before the → and a figure after

Figure 24 Specification for Correcting Certainty

8. Example System

The following is a partial example of an expert system which provides decision making information on choosing different types of employment.

Table 2 Example fact table

< Facts >

1. Employment location < Tokyo area/place of birth/other > is preferable
2. Type of employment < Primary sector/secondary sector/tertiary sector > is preferable
3. Size of company < Major corporation/medium sized company/either > is preferable

4. Starting salary < high/either high or low > is preferable
5. Your personality < extroverted/introverted >
6. Aptitude < planning/desk work >
7. Hobbies < sports outdoor/reading indoor >
8. Work < general/specialized > is preferable
9. Track record < outstanding/ordinary/poor >
10. Physical strength < outstanding/ordinary >
11. Pleasant appearance < yes/no >
12. Previous employment experience < yes/no >
13. Periods of unemployment < none/up to one month/up to three months /up to one year/greater than one year >
14. Special skills < baseball/rugby >
15. Recommended sector < secondary sector/tertiary sector > is best
16. Recommended type of employment < sales/office work > is best
17. Recommended type of work < focus on work/focus no time off > is best
18. Recommended type of work < regular hours/free to choose time of working > is best
19. Recommended form of testing < interview/paper test > is best
20. Company to aim for < company A/company B/company C/company D/company E/company F/none >

Table 3 Example of rules

< Rules >

1. If the area of employment is < Tokyo area > and track record is < excellent > then company to aim for is company A
2. If size of company is < small－medium > and track record is < ordinary > then company to aim for is company C/company d/company E
3. If the area of employment is < place of birth > and starting salary is < high > then company to aim for is NONE
4. If the type of employment is < secondary sector > and area of employment is < other > then company to aim for is company E

5. If the type of employment is < tertiary > and starting salary is < either > and area of employment is < Tokyo > then company to aim for is company A

< Example of database for company to aim for >

Company name	Lo – cation	Sector	Size	Starting salary	Track record	Willingnes
A	Tokyo	Tertiary	small/medium	Hight	Important	Important
B	Tokyo	Secondary	Large	Ordinary	Important	none
C	Local	Secondary	small/medium	Ordinary	Important	none
D	Local	Tertiary	small/medium	Ordinary	none	Important
E	Other	Secondary	small/medium	Hight non	none	none
F	Other	Primary	small/medium	Ordinary	none	Important

Figure 25 Characteristics of each company

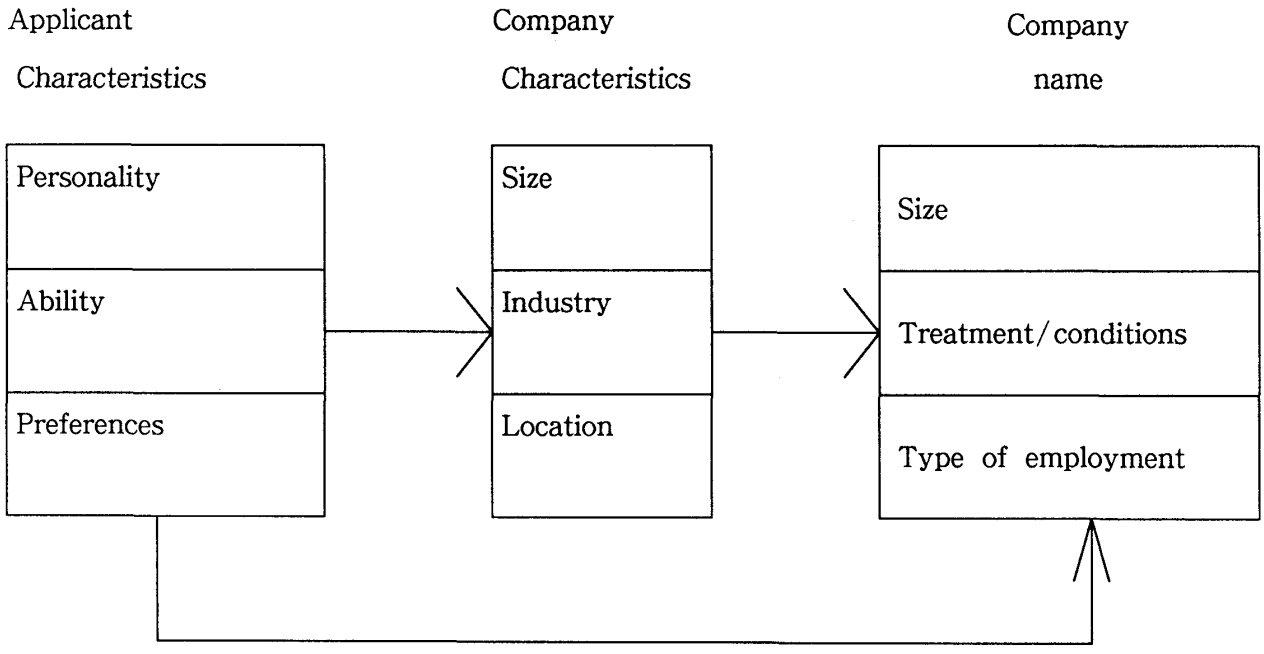

Figure 26 Structure of rules

9. Conclusion and discussion

This paper puts forward a system specification for developing a knowledge system, based on the example of an expert system. The two major components of this proposal are 1) a method of expressing facts and rules, and 2) a method of analysis using knowledge system charts.

However, as there are many ways of representing knowledge there are many questions which are still left unanswered to say whether or not this is an effective method of system design. For example, with knowledge systems using frame reasoning or blackboard methods, what are the common areas of development in terms of developing knowledge systems. Also, how should we deal with the areas of development which differ from the development process put forward here. We must wait for further results from research on knowledge representation. But, at the same time we must experiment with constructing conceptual forms for knowledge itself from the standpoint of developing practical and workable knowledge systems.

Source of Reference

(1) S.M.Weiss,C.A.Kulikowski, 『A Practical Guide to Designing Expert System』, rowmon & allaheld,1982

(2) Yu Nanjyo, *EDP sisutemu Sekkei nyumon*, pub.ohmusya,1980

(3) Hidetoshi Takahasi, *jyouhoukagaku no ayumi*, pub.iwanami syoten, 1980

(4) Setuo oosuga, *detabesu to tisiki sisutemu*, pub.ohmusya,1989

(5) Frederick Hayes－Roth,『Building Expert System』, Addison－Wesley,1983

(6) Daisuke Miura, *sisutemu no bunseki to sekkei*(1), pub.ohmusya, 1973

(7) Haruki Ueno, *tisiki kougaku nyumon*,ohmusya,1985

(8) Masahisa honda,Tadatosi Yamaguti, Nobuhuko Seike, *gyouseisi detabesu no kaihatu*, pub.gyouseijyouhou kenkyujyo, 1983

(9) Nobuhiko Seike, *souzouseigihou handobukku*, pub.bijinesuripoto,1981