Acquisition of terminological knowledge using database design techniques

Christoph F Eick Peter C Lockemann

Fakultat fur Informatik
Universitat Karlsruhe
Postfach 6380
D-7500 Karlsruhe 1

Abstract

One of the most difficult problems in knowledge base design is the acquisition and formalization of an expert's rules concerning a special universe of discourse In most cases different experts and the knowledge base designer himself will use different terminologies, and will represent rules concerning the same objects in a different way Therefore, one of the first steps in knowledge base design has to be the construction of an integrated, commonly accepted terminology, that can be shared by all persons involved in the design process This design step will be the topic of the paper The paper proposes concepts, methods and tools to support the extraction, integration, transformation and evaluation of terminological knowledge that are based on database design techniques and discusses the possibilities and limitations of automating these
keywords and phrases knowledge based systems, knowledge base design, database design, conceptual modelling, semantic modelling, terminological knowledge acquisition, knowledge integration, design automation

1 Introduction

The economic order of the modern world is characterized by a scarcity of natural resources and an abundance of highly skilled but also highly-paid labor forces in the traditionally industrialized countries Consequently, producing industries have a tendency to move their activities to the less well developed countries To fill the void, the industrialized countries have to turn to industrial enterprises with a high rate of value added to the raw products To sustain their economic rigor, these countries must maintain the high qualification of their labor force as well as a large body of technological expertise from which the labor force can draw its know-how

The present paper is concerned with the latter, i e the management of expertise or to use a modern term, of knowledge bases The importance of knowledge bases for both easy access and versatile manipulation by compiler based means, has long been recognized and particularly well documented in the Japanese Fifth Generation Computer Project (fig 1)

A knowledge base can be considered to describe propositions of two kinds (fig 2)

1) Propositions which have a high stability and generality for the universe of discourse (UoD) of a given application We shall refer to these propositions as **rules** It is usefull to distinguish between rules that hold for the terminology used to describe the UoD (**terminological rules** often also called intensional knowledge) and rules that describe the general behaviour of the UoD **(behavioural rules)** Obviously behavioural rules make use of the terminology specified in the terminological rules for the UoD
2) Propositions that pertain to a specific state of the UoD and/or may change with time We call this propositions **facts**

Knowledge bases have been an active field of research over the past years, a fair number of such bases have been established as part of so-called expert systems [McD 83] One of the most vexing problems has turned out to be capturing the knowledge of human experts in a form that can be put into a machine To be successful one would have to employ some strict discipline that extracts the knowledge from the expert in a stepwise fashion, translates it into a form that facilitates the communication between the knowledge base designer (or **knowledge engineer** [FEI 77]) and the expert, allows the expert to validate or correct the results, and finally puts it into a form suitable for storage in and manipulation by a computer The development of such a discipline (called **knowledge acquisition)** is the specific concern of this paper

A number of difficulties may be expected in the course of the task

* **Experts** for the specific UoD may have difficulties in formalizing their knowledge
* One important goal of a knowledge base system is to make use of the knowledge located in different information sources, for example from different experts However, very often different experts use different terminology, which implies that they often represent the same rule in a different way

* In most cases the knowledge engineer will not be sufficiently familiar with the terminology of the application area Therefore, one difficult problem of the knowledge engineer is the acquisition of the terminology he needs in order to understand the rules which are informally given to him by the experts
* The rule set as provided by an expert generally is incomplete in the sense that the expert often makes implicit use of some trivial rules (for example that "every woman is a human being") Often the expert is not aware of his implicit rule reference and will not formalize these rules
* Lastly, we have the group of the **end users** of the knowledge base system, who normally are not very familiar with the application area of the knowledge base system and are not familiar with the formalized language used for the specification of the rule base as well This implies that after the design of the rule base, costly training of the end users has to be performed

We conclude from the discussion that a strategy for designing the terminological and behavioural rules of a knowledge base should follow a number of steps

1) Determine for each expert involved his/her terminology
2) Based hereupon, obtain from each expert the behavioural rules, usually in an informal fashion
3) Integrate the various terminologies into a commonly accepted terminology
4) Coalesce the various sets of behavioural rules into a single, commonly accepted set of formalized rules

Fig 3 illustrates the **strategy** (note that the knowledge engineer will usually contribute his early understanding of the application area)

Despite the importance of knowledge acquisition, the subject is still treated very lightly in the literature Weiner [WEI 81] has proposed a manual procedure to derive the structure of a database from natural language inputs In [ADE 83] a form technique is proposed, that may be used to facilitate the acquisition of terminological knowledge In [ELNA 84] and [BAT 83] some special problems for knowledge integration have been treated [NAGA 82] propose how knowledge integration should be organized in large organisations
Other papers ([DAV 79], [GRO 83], [MIT 83]) have proposed some general techniques for knowledge acquisition However, they do not take into account the specific character of terminological knowledge [WED 80] has stressed the importance of terminological knowledge for a good information system design

It is often overlooked in the general discussion on knowledge bases that classical database design has concerned itself with the acquisition of terminological and behavioural rules for many years, albeit under headings such as "semantic modelling" or "conceptual modelling" Our paper will report on a project (**ANNAPURNA**) that aimed at providing a computerized environment for semi-automatic database design, covering all phases from knowledge acquisition from the application experts all the way to generating an optimal database schema for a given database management system We feel that particularly the methods and tools for the early phases are versatile enough to be applied towards the more general task of knowledge base engineering

In the paper we shall concentrate on the phases concerned with acquiring the terminological rules In particular, we shall sketch techniques with which terminological knowledge can be extracted, transformed, evaluated, integrated and refined Both opportunities and limitations for automating these techniques will be discussed In doing so we need a more precise definition of what we mean by "terminology" We shall use this notion in the sense of explaining a given term by means of other terms, much like this is done in a thesaurus or in the Oxford dictionary In other words, the terms of a particular application area are studied in the context of other terms with which it undergoes relationships

The paper is organized as follows In section 2 a formalism for the description of the terminological knowledge, S-diagrams, is introduced, and the design phases of our methodology will be discussed Section 3 to 5 treat the different phases of the knowledge acquisition process Section 6 will summarize our experiences with applying the methodology in several experiments

2 Assumptions of the acquisition process

2 1 S-diagrams

For the description of the terminological knowledge we use a graphical data model called **S-diagram** which was influenced by the binary relation model ([ABR 74]) and by SDM ([McL 78]) We shall illustrate the concepts of S-diagrams using the following example dealing with the connection of rooms by doors (fig 4)

Using S-diagrams it is possible to specify **classes** (in the example, connect, room, door and physical_object are classes), **subclass connections** between classes (for example both rooms and doors are physical_objects), and **attributes** (from_room, to_room and by_door are attributes) which describe properties of members of classes An attribute has a **domain** class and a **range** class (for example by_door has the domain class connect and the range class door) An attribute may assign to a member of the domain class zero, one or more members of the range class, and each member of the range class may occur as an attribute value zero, one or more times
These cardinalities associated with an attribute may be restricted using the **labels** multivalued, unique, optional and onto
Let att be an attribute with domain class K1 and range class K2, n1 the minimum and n2 the maximum number, respectively, of members of K2 which may be connected via att to a member of K1, m1 the minimum and m2 the maximum number, respectively, of references from members of K1 to a member of K2 via att The pair (n1, n2) specifies the cardinality of att and (m1, m2) the cardinality of the converse of att, called att^{-1} The labels multivalued and optional define restrictions on (n1, n2) as follows (* represents "many")

(n1, n2)-combination	label combination
n1=0, n2=*	multivalued, optional
n1=1, n2=*	multivalued
n1=0, n2=1	optional
n1=1 n2=1	----- (no label, default)

Simlary the labels onto and unique define restrictions on (m1, m2) as follows

(m1, m2)-combination	label combination
m1=0 m2=*	----- (no label, default)
m1=0, m2=1	unique
m1=1, m2=*	onto
m1=1, m2=1	unique, onto

The labels and subclass connections will be represented in S-diagrams using the graphical symbols presented in fig 5

In the S-diagram of fig 4 the attribute by_door carries the label set {onto}, hence n1=1 n2=1, m1=1, m2=* expressing the following semantics there are no doors that do not connect to some room (because of the label onto) a connection may only have one door, and if there is a connection there must be a door for this connection (because there is neither label optional nor multivalued), and one door may be involved in different connections (because there is no label unique) (As a matter of fact, m2=2, however S-diagrams do not have the full expressive power of cardinalities)

Since informal knowledge may often be incomplete, at least in the initial acquisition phases, it is useful to relax the formal rules for S-diagrams and allow that

- attributes have no name, no domain class or no range class,
- label information is omitted

2 2 Phases of the acquisition of terminological knowledge

The premise underlying our approach is that the prime purpose of knowledge base sytems is to produce answers to queries put by the end users to the system The rules and facts stored in the knowledge base are the input from which the answer will be created

Hence one way to extract the pertinent knowledge from the expert is to find out from him/her which queries he/she may put to a supposedly expert system, and what the general rules are that he/she would expect the system to be aware of in answering the queries In this early phase of the acquisition process one can clearly not expect the expert to be very formal Rather, queries and rules will be expressed in natural language

Consequently, the first design step will try to extract the terminological knowledge from queries and rules that have the form of natural language expressions The knowledge obtained in this step will be described using (possibly incompletely specified) S-diagrams
We call a collection of persons who share or who are assumed to share the same terminological knowledge a **user group** In the second step a complete, collective S-diagram has to be derived for each user group from the individual S-diagrams (obtained in the first step) Furthermore, in order to prepare the subsequent steps, the S-diagrams received for the different user groups will be brought into a canonical form by applying S-diagram transformations
In a third step the S-diagrams of all the user groups will be integrated into a single S-diagram which will then be a description of the terminology used in the whole universe of discourse The major difficulty to be expected is that now we shall have to deal with divergent terminology Hence this step is some kind of 'consensus building process'
In a fourth step the integrated S-diagram will be enhanced to conform to certain quality measures This will again be done with the help of S-diagram transformations

In section 3 the first step will be introduced Section 4 discusses the evaluation and transformation of S-diagrams to be used in steps 2 and 4 Section 5 addresses the issues arising in step 3

3 Extraction of terminological knowledge

As mentioned above we assume that in the first step rules and queries will be provided informally, using natural language For example

> 'Can box '17 be pushed from room 194' to room 444 ? '

may be a description of a query q1 and

> if room *X and room *Y are connected by door *D **and** a physical object *O is smaller than the door *D **then** *O can be pushed from room *X to room *Y "

(strings beginning with '*' represent universally quantified variables) may be a description of a rule r1, which can be helpful to create an answer to the previous query

We now try to extract the terminological rules contained in these natural language specifications and put them into the form of (possibly incompletely specified) S-diagrams

During this extraction process the following problems must be solved

1) Terminological rules contained in a natural language specification have to be separated from other kinds of information
   From query q1 we would like to extract that "boxes can be pushed from one room to another room and that boxes and rooms can be identified by numbers Conversely we are not interested in the information that "a specific box with number'17' can be pushed from a room which has the number '194 to a room which has the number '444' ", because this would be a description of a fact
2) Because of the ambiguities inherent in natural language, it is sometimes necessary to assign more than one S-diagram to a natural language specification
   For example, from the sentence fragment " 'X' supplies companies with TV sets " one cannot infer whether TV set has to be connected to the company which is being supplied ("X supplies only companies owning TV sets ') or to the supplying activity ("X supplies TV sets")
3) In some cases it is not possible to construct an S-diagram from a natural language specification, because some knowledge necessary for the S-diagram construction may not be contained in the natural language sentence to be analysed (an example of this situation will be given below)

Because of the large number of queries and rules to be examined one would like to solve at least the unproblematical extraction cases by automated tools, leaving only the difficult cases of human interaction Of course, those cases would have to be identified by the tool

A tool, AISCHYLOS, with these properties was developed as part of the ANNAPURNA project It uses a set of (presently) 43 heuristic rules to generate an S-diagram from the grammatical structure of a natural language sentence

To give the flavor of the rules, consider the following example of a rule that describes the treatment of a class of sentence predicates in the S-diagram generation process

standard predicate rule
context of rule application treatment of sentence predicates
precondition The main verb associated with the sentence predicate is not be, have 'consist and the predicate does not include conditional verbs (like may can)
effect A class named by the infinitive form of the main verb of the predicate will be constructed All classes constructed for the subject and the objects of the sentence will be connected by an attribute to this class The attributes will be named by the the name of the classes to which they refer In the case of a preposition in front of an object the corresponding attribute name will be prefixed by the preposition

Remark The upper part of the S-diagram (fig 7) for rule r1 has been constructed by applying the standard predicate rule for the predicate "are connected"

Besides constructing an S-diagram or parts of it, rule processing builds a dictionary to be used for linguistic processing (entries contain e g singular nominative form, superclasses,) If during rule processing a new term is encountered that cannot be found in the dictionary, the user is queried by the tool in order to define the new dictionary entry Similarly, the tool interrogates the user in case of ambiguities or incompleteness

Complex sentences are broken down into sets of smaller sentences For example the sentence describing rule r1 is divided into the sentences "room *X and room *Y is connected by the door *D , "a physical object *O is smaller than the door *D" and "*O can be pushed from room *X to room *Y ', which then form separate inputs to the S-diagram construction part of the tool In order to illustrate the dialogue of the tool with its user we shall use the analysis of the sentence a physical object *O is less than the door *D

Tool Does 'physical' attached to 'object' describe
1 a property of 'object'?
2 a compoud class of objects named physical_object ?
3 some irrelevant detail of 'object' ?
User 2
Comment The answer 1' is the normal case, as in the phrase "pink car" because the color would be regarded as a property of car This will afterwards be represented by an attribute definition in the new S-diagram
Tool *O describes a property of 'physical object', how is this property called ?
User Objectnumber
Tool *D describes

Tool The comparative 'less' compares a common property (set of properties) of physical_object' and door, how is this property (set of properties) called ?
Comment Here we have an example where we need additional information for the S-diagram construction
For an expert the meaning of less in the context of doors and physical_objects is obvious However, for the use in a knowledge base system this knowledge has to be formalized, that is the information needed for the evaluation of the predicate less has to be specified
User Length, Breadth
Tool How will the values of Length be represented ?
User FIXPOINT (9 2)
Tool How will the values of Breadth

Tool Does there exist a common super class of 'physical_object and door, which has the attributes length, breadth ?
User Physical_object
Tool From your last answer I can infer that 'door is a subclass of physical_object', is that true ?
User yes

In summary AISCHYLOS constructs from rule r1 and query q1 the S-diagrams of fig 6 and 7

In general S-diagrams created by from AISCHYLOS are incomplete since the attribute labels can only be determined under certain circumstances such as the use of the deterministic article (label "unique"),plural (label multivalued"),

AISCHYLOS was used in several classroom experiments as part of designing a network database schema From these we can draw the following conclusions
- A surprising result was that it was possible with a relatively small set of about 40 heuristic rules to construct correct (see section 4) S-diagrams in more than 95% of the cases examined
- A fairly heavy price must be paid in terms of software development costs AISCHYLOS is written in Pascal and contains about 8500 lines of statements which may be categorized as follows

scanning and parsing of natural language	2500 lines
S-diagram generation	2800 lines
user communication and internal administration (dictionaries,)	3200 lines

- One important aspect of our experimentation was to test the user acceptance of such a tool Initially, the tool will ask its user a large number of (sometimes very simple) questions, because its knowledge base about the UoD is rather small This may discourage a highly educated and highly paid expert

Hence one of the most critical problems in the design of such a tool is the development of good user communication strategies that on the one hand avoid misunderstandings by the user of the system's questions and answers and on the other hand minimizes the number of question put to the user Some useful techniques to solve these problems have been proposed in projects for natural language interfaces to computers ([PIL 84], [HOP 83] [MAP 83])

4 Evaluation and transformation of S-diagrams

One absolute requirement of a good knowledge base design is that all queries formulated for the planned system may be satisfied This implies that **all** propositions (facts and rules) relevant for this query satisfaction process should be representable in the knowledge base to be designed
Consequently we call a terminology **correct** if **all** propositions relevant for the query satisfaction of the knowledge base system may be described using this terminology That is all classes, attributes and subclass relationships necessary for the description of the specific UoD have to be defined in the S-diagram

Correctness does not imply uniqueness of diagrams however In fact for a given UoD a large number of

correct S-diagrams can be found For example our room/door/connection example may also be described using the S-diagrams (2) and (3) instead of the S-diagram (1) (fig 8)

The example brings up the following questions

(Q1) Suppose the original S-diagram (1) is known to be correct, can one deduce whether the S-diagrams (2) and (3) are also correct?

(Q2) If several S-diagrams are correct for a UoD, for example (1), (2) and (3) which of these is "optimal"?
Consider for example, that one expert is found to use terminology (1) and another expert terminology (3) Which S-diagram should be chosen during the integration step to describe the combined terminology for both experts ?

(Q3) How can the optimal S-diagram be obtained from some initial, correct S-diagram ?

Satisfactory answers to these questions require a careful theoretical analysis The theoretical framework has been reported elsewhere ([EICK 84b], [RAU 84], a briefer outline may be found in [EIRA 85]) The basic idea is to introduce the notions of "generalized functional dependencies" and "generalized existence dependencies", both combined under the notion of "X-dependencies" The class of X-dependenens is an upper bound for the rules that can be expressed in an S-diagram, that is every (terminological) rule expressed in an S-diagram is contained in the class of X-dependencies
Using this framework transformations of S-diagrams may be regarded as transformations of X-dependencies a specific transformation maps the set of the X-dependencies expressed in the original S-diagram to a set of dependencies (which are also hoped to be X-dependencies, however, this property cannot be generally shown for powerful transformation systems (sets of S-diagram transformations)) that should hold for the transformed S-diagram

Given a set of transformations on an S-diagram, one must introduce some kind of quality measure that, in principle, allows to compare the results of the transformations and choose the optimal one

The details are beyond the the scope of this paper We shall, however, give a summary of the premises and the results to demonstrate the general approach

Question (Q1) may be rephrased as "Given the correct S-diagram (1), is there an **information preserving transformation** that constructs S-diagram (2) or S-diagram (3) from it ?" (A transformation is called information preserving if it maps a correct S-diagram to a correct one)

We shall demonstrate the basic idea for the example of fig 8 If the connection relationship of rooms is symmetric for a specific door (that is every door connects rooms in both directions), the transformations from S-diagram (1) to the S-diagrams (2) and (3) are information preserving
However, if the above symmetry relationship does not hold (that is, some doors may only be used in one direction) the transformation from S-diagram (1) to S-diagram (3) is no longer information preserving, because the following two propositions
p1) door d1 is a connection from room 2 to room '14'
p2) door d1 is **no** connection from room '14 to room '2'
are not expressible, when using terminology (3)
Note that the two cases just discussed differ in that in the first case a condition has been added (symmetry of the relationship) that is not directly being expressed in S-diagram (1)

Proceeding now to question (Q2) we need a quantitative quality measure for S-diagrams Clearly, such a measure should ignore the more or less coincidental naming conventions in an S-diagram and concentrate on the **structure**(classes attributes and subclass relationships) of an S-diagram A quantitative quality measure for S-diagrams has been defined in ([EICK 84a] [EICK 84b]), which incorporates the following quality factors

(Qu1) The **complexity** of the S-diagram to be evaluated number of classes attributes labels and subclass relationships in the S-diagram
For example S-diagram (2) is less complex than the S-diagram (1) in that sense

(Qu2) The **expressiveness** of the S-diagram to be evaluated the proportion of terminogical rules expressed in the S-diagram
For example, S-diagram (3) does not express the rule that a door connects exactly two rooms (n1 of the attribute connects should have the value 2) This rule is expressed in the S-diagrams (1) and (2) (in S-diagram (1) because the attributes from_room and to_room have neither the label multivalued nor optional) Consequently, the expressiveness of the S-diagrams (1) and (2) is higher than that of S-diagram (3)

(Qu3) The **normalizedness** of the S-diagram to be evaluated The idea underlying this quality factor is, that if two different human beings describe the same UoD using two S-diagrams S1 and S2 which are equally good in the sense of our quality measure then the structure of S1 and S2 should be equal or very similar The importance of normalization will be discussed in section 5

An important result in [RAU 84] is the existence of an algorithm that allows a completely automatic evaluation of S-diagrams relative to this quality measure The most difficult problem that had to be solved by the algorithm was the evaluation of the expressiveness of an S-diagram, a test procedure had to be constructed that decides, whether a given terminological rule is directly expressed in an S-diagram or can be inferred from the rules directly expressed in the S-diagram

Concerning the third question (Q3) a system of S-diagram transformations has been developed that allow to enhance the quality of S-diagrams The theoretical properties of the transformation system are discussed in detail in [EICK 84a] [EICK 84b],[RAU 84]

Unfortunately there is no one-step procedure for selecting the transformation to the optimal S-diagram Rather, a number of transformations must be applied — selected perhaps by heuristic rules — and the results compared with regard to quality Clearly, therefore automation of the transformations is of vital importance

Our results concerning the transformation and evaluation of S-diagrams can be summarized as follows
* The evaluation of an S-diagram concerning its structure may be completely automated
* The test, whether the application of a specific transformation (of our transformation system) to an S-diagram is information preserving may be performed almost entirely automatically (in some specific situations help by the human designer is needed because of missing information or because of some decidibility problems, that have not been solved so far)
* The calculation of the transformed S-diagram (its labels,) received by a specific transformation can be performed completely automatically by a computerized tool

5 Integration of terminological knowledge

5 1 Integration problems

When different human beings refer to the same UoD they will very often use a different terminology If these human beings (user groups) have to cooperate in a project, they must be able to understand one another, that is, a correct, understandable and acceptable terminology (communication base shared by all users) has to be created by the **integration** of the terminologies used by the different user groups Normally, this integration activity will be performed by verbal discussions between the user groups

In this paragraph we will discuss the problems of integration of terminological knowledge and how far these problems can be solved by computerized tools In the following, we assume that the terminological knowledge of the user groups has been decribed using S-diagrams

The integration process (of S-diagrams) can be subdivided into three steps

(St1) The conflict situations between user views have to be **detected**

(St2) The conflicts encountered in step St1 have to be **resolved** and an **integrated S-diagram** has to be **constructed**

(St3) The correspondance between the views of the individual user groups (views) and the integrated schema has to be recorded, that is **view definitions** have to be **specified** that allow the reconstruction of information relevant to the user groups from the information represented using the terminology of the integrated S-diagram

5 2 Conflicts and their resolution

Turning first to steps (St1) and (St2) the following conflict situations must be treated

(T1) Homonyms for class and attribute names must be removed such that unique objects have unique names (name unification)

(T2) Contradictory specifications in different user views regarding subclass relationships, attribute range classes and attribute cardinality restrictions must be resolved

(T3) Common objects shared between different user views but named differently (synonyms) and other redundancies must be detected and resolved

If (T1) is not properly dealt with, the result of the integration will be an incorrect S-diagram (differences in rules are not always expressed) Not solving (T2) clearly means that integration may not take place at all (which of the several alternatives should one choose?) Missing (T3) is at least tolerable but gives rise to uncontrolled redundancy

To give a better feeling for these kinds of conflicts and the ways how to resolve them we start out with a simple example Concider the following two user views B1 and B2 of the same UoD with the only difference being the range classes of the attribute length (fig 9)

Here we have a contradictory range class definition for the attribute length because (in the integrated S-diagram) the attribute length may only have one range class (and not two)

The detection of conflicts of this kind, and similary of other contradictions between S-diagrams such as contradictory attribute cardinalities or subclass specifications can be performed automatically The strategies for resolving the conflicts are different, of course, depending on the different possible causes of the contradiction

1) The integrated range class could be the union of the range classs specified by the user groups (in the example this would be FIXPOINT(9,4)) For example user group B1 is interested in large and user group B2 in small physical objects
2) Length in B1 and B2 could be a homonym In this case we do not have a contradictory range class definition but an undetected attribute name homonymity concerning length For example length could denote physical length in B1 and wave length in B2
3) The class name physical_object could be a homonym relative to B1 and B2 In this case we do not have a contradictory range class (because the attribute length belongs to different classes) but an undetected class name homonymity
4) FIXPOINT(9 2) and FIXPOINT(6,4) could be synonyms (which makes little sense for this example) In this case we have no contradiction but an undetected class name synonymity

Clearly, the cause of the conflict itself cannot be decided by a computerized tool (if the tool has no additional knowledge about the inter-usergroup relationships) Therefore, the resolution of contradictions may not be completely automated However, a computerized tool may assist the designer by displaying to him/her all possible resolution alternatives from which he/she chooses the one most appropriate to the given situation

In [ELNA 84] more general treatment of situation 1 is given, however the naming problems (causes 2-4) are ignored in the papers We believe that, for the prize of the human interaction, our approach is more generally applicable particularly in view of the fact that all four cases will occur together in practical situations, so that the interdependencies between them must be taken into account

5 3 Identification of conflicts

In order to choose the appropriate resolution strategy, the designer must first recognize all potential candidates for synonymity, homonymity and subclass connections among two ore more user views to be integrated Consider however, that for S-diagrams with realistic sizes of 100 classes, the designer would have to inspect nearly 5000 potential synonym- and subclass candidates

Clearly in general he/she will limit him/herself to a small number of these candidates, because a test of all candidates will be extremely time consuming and expensive But how can he/she be sure to select the most relevant candidates? A computerized tool that preselects candidates which have a high probability of having a subclass/synonymity relationship will be of great help in the design process

Two concepts are important for the solution to this and other problems of automatic conflict situation detection between different user views

The first concept is the **normalization** of S-diagrams before they are integrated Normalization is achieved by observing certain rules during the semi-automatic process of constructing S-diagrams

1) Application of naming rules
   For example "class names may only be substantives in nominative singular form or verbs in infinitive form" is a naming rule which excludes a situation in

which in one user group a class is named 'has_been_connected and in a second this class is named connects This will simplify the detection of the synonymes between user groups

2) Standardization of structure
An initial S-diagram will be transformed into a good S-diagram relative to the quality measure mentioned in section 4 before the integration is performed This structural normalization process will reduce the differences among the S-diagrams to be integrated If, for example one user group has used terminology (1) of figure 8 and a second has used terminology (2) the normalisation process will guarantee that the normalized S-diagrams to be integrated either have the structure (1) or the structure (2)

One advantage of this approach is that many problems of the detection of complex redundancies between user groups (like the redundancy between the S-diagrams (1) and (3) before the normalisation process) will be reduced to the detection of synonyms and subclasses between user groups

The second concept is the application of several **similarity measures** for the selection of probable synonym subclass and homonym candidates between user views, which will be used for the computerized analysis of inter-S-diagram relationships Similarity measures are based on different matching factors that will be evaluated in the **environment** of the classes K1 and K2 for which one of the above cases has to be tested The environment of a class K is

* the set of attributes for which K is domain or range class
* the set of classes, which are a subclass or a super class of K, or the domain or range class of an attribute with range or domain class K, respectively

Take as an example the similarity between classes that is based on a class inclusion K1 achieves a high score with regard to a class K2 if it "appears" to be a subclass of K2 To test the hypothesis, four matching facts are considered The score is high

- if a large number of attributes of the environment of K1 are contained in the environement of K2
- if many classes of the environment of K1 are contained in the environment of K2
- if there is a high similarity between names used in the environment of K1 to names used in the environment of K2
- if we assume that K1 and K2 are synonyms or K1 is a subclass of K2 there should only be a very small number of contradictions between the environments of K1 and K2

In order to illustrate the importance of the detection of synonyms, subtypes and homonyms in knowledge base design we return to our previous example If we integrate the S-diagrams of figure 6 and 7 without trying to detect subclass relationships between the classes to be integrated we will run into the following problem We cannot use rule r1 for answering question q1, because the terminological knowledge that box is a subclass of physical_object has not been specified

The automatic calculation of the matching factor of box for physical_object will take into account the following information

- can_push and INTEGER are also in the environment of physical_object
- boxnumber has a textual similarity to objectnumber
- no contradiction occurs if box is subclass of physical_object

In evaluating the matching factors a computerized tool should attach to the subclass connection of box and physical_object a high score so that this conflict candidate will not be eliminated by the automatic preselection process Later it will be presented to the user who has to decide whether the subclass connection really holds in the UoD or not

5 4 View definition

The last problem to be discussed is the automation of step (St3) If for example one user group represents its rules using represention (3) of fig 8, and the integrated S-diagram uses representation (1), the user group will still insist on using its own terminology (3) (and not (1)) This will be true not only during the later everyday use of the knowledge base, but also during the subsequent acquisition of further expert knowledge and the consequent augmentation of the integrated S-diagram

View definition can be considered a by-product of the steps discussed in sec 5 2 and 5 3 In every case, in which the representation of a user view has to be changed because of contradictions, redundancies or homonymities to other user views, we log the transformations applied to user view S-diagrams during the conflict resolution

6 Conclusion

The methods and the tools described in this paper were initially developed in the context of a computer-based database design environment and extensively tested in a database laboratory with junior students with only very crude textbook knowledge on database design In parallel a number of diploma theses were done refining the methods and tools, examining the theoretical framework and implementing the tools with the overall intention of pushing automation of database design to its limits The results in the paper can, therefore, be presented with some confidence on our parts

We have shown that the acquisition of terminological knowledge namely its extraction from natural language statements its formalization via S-diagrams, its integration and its transformation into a more advantageous form can all be supported by automatic, computerized tools On the other hand, because of incomplete presentation of knowledge, idiosyncrasies of particular user environments or ambiguities and redundancies full automation of the design process is fundamentally impossible As a consequence, the tools must be able to interact with the designer, and, hence careful consideration should be given to the user interface of the tools In particular, 'intelligent" user interaction strategies must be developped to raise acceptance of tools An interesting result of our project was that the acquisition of the corresponding information (needed in the heuristic rule application process) was the most critical part of the tool design and not the S-diagram construction process itself

We have tried to improve the process of knowledge integration by the use of quality **and** similarity measures Using a quality measure the S-diagrams to be integrated will be put to a canonical form (by applying S-diagram transformations) This normalization process

guarantees that entities which have structural similarities (concerning the rules that hold for them) will be described in the same way and entities which are structurally different will be represented in a different way Assuming that all S-diagrams to be integrated have the same canonical form the detection of synonyms homonyms and subclasses will be performed using similarity measures
It is important to state that, in our opinion, the use of similarity measures without normalization prior to integration (as it has been proposed in [BAT 83]) will not be of great help to the designer, because the same proposition may be represented in a large number of different ways which will make it very difficult and complex to find similarities between nonnormalized S-diagrams Using the concept of a normalized S-diagram the number of cases in which the same proposition will be described in different ways will be drastically reduced, resulting in the simplification of the integration process

The reader will have noted that we never precisely stated what our quality measure was or, consequently when an S-diagram was "advantageous" [EICK 84b] introduces a measure based pretty much on the quality factors (Qu1) (Qu2) and (Qu3) in section 4, appropriately weighted However it is still too early to state with confidence whether the measure is entirely suitable, or by which environmental factors it may be influenced Generally speaking, these S-diagrams will score best that have small complexity and large expressive power, that is a good (integrated) terminology has to be simple and expressive

Another advantage of our methodology is that it is based on natural language specifications This guarantees that comprehensible names will be used for the S-diagram construction Furthermore, many problems that may be encountered in the design process may be solved by going back to the natural language specifications (for example in the case of a potential class name homonymity, the designer may be asked by the tool "Does the term ' used in rule 5 " " and rule 334 " " refer to the same class of objects?)

We claim that a semi-automatic design process such as the one described in the paper is more economical than an entirely manual process and yields much better integration results at least for larger size UoDs However, more well-controlled experimentation is necessary to validate this hypothesis

Lastly our recent exposure to work in the area of expert sytems has convinced us that our work, originally developed for database design has much wider application in the general area of knowledge base construction

Figures of the paper

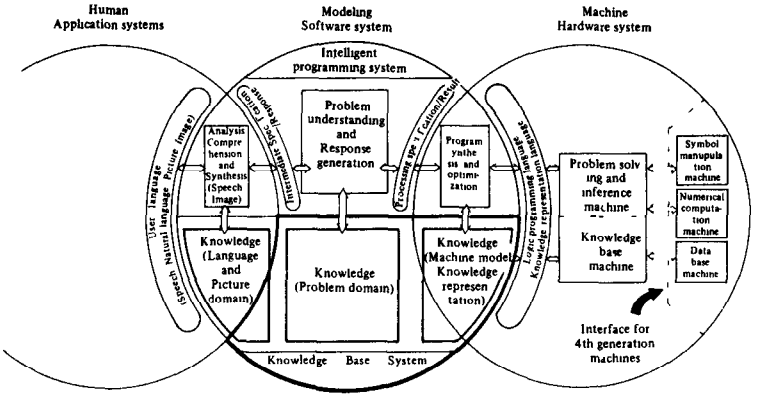

Fig 1 architecture of a fifth generation system taken from [MO 82]

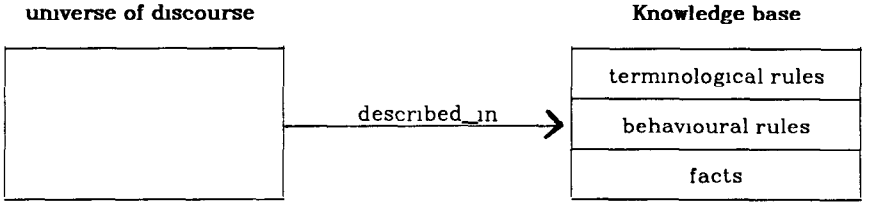

Fig 2 contents of a knowledge base

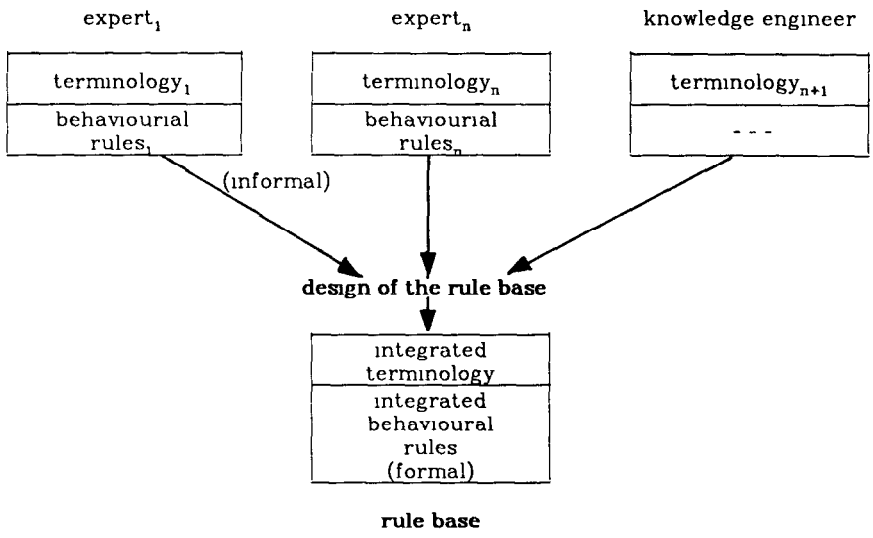

Fig 3 rule acquisition in knowledge base design

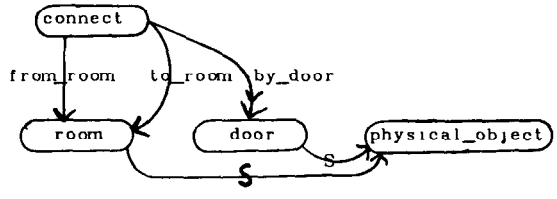

Fig 4 example of an S-diagram

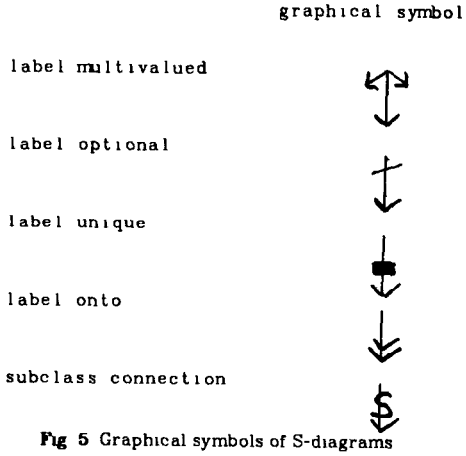

Fig 5 Graphical symbols of S-diagrams

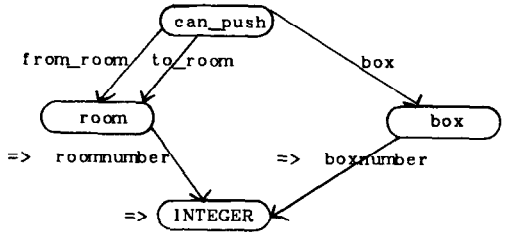

Fig 6 S-diagram constructed for question q1
(=> indicates that the class attribute or subtype definition has been determined by user interaction and not automatically)

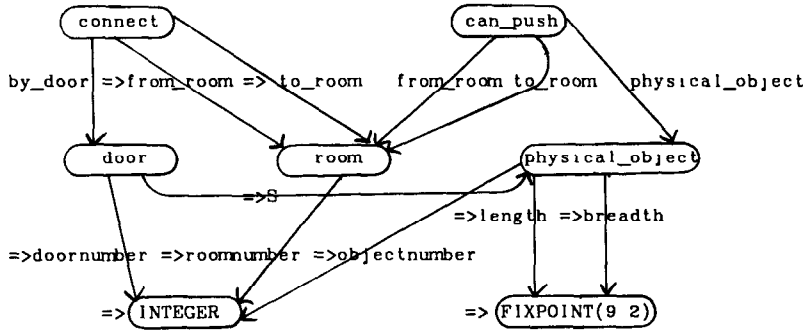

Fig 7 S-diagram constructed for rule r1

(The attribute names from_room and to_room have been generated interactively because of the application of the standard predicate rule had lead to an ambiguity by introducing the attribute name room twice which had to be resolved by an user interaction)

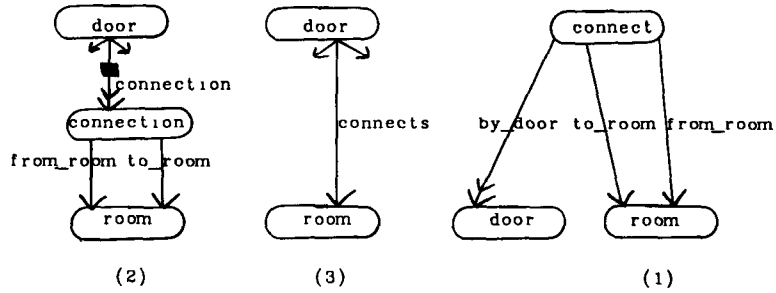

Fig 8 several candidate S-diagrams for a UoD

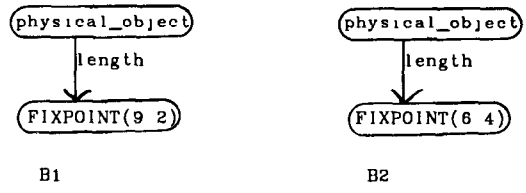

Fig 9 example of a contradiction between user views

Literature

[ABR 74] Abrial,J R
"Data Semantics'
Proc IFIP TC2 Conference North Holland (1974)

[ADE 83] De Antonellis,V Demo,B
"Requirement collection and analysis"
in Cery S "Methodology and tools for data base design"
North Holland (1983)

[BAT 83] Batini,C ,Lenzerini M ,Moscarini,M
"View integration"
in Cery,S "Methodology and tools for data base design"
North Holland (1983)

[DAV 79] Davis R
' Interactive transfer of expertise"
Artificial Intelligence, p120-157 (1979)

[EICK 84a] Eick,Ch ,F
"From natural language requirements to good data definition
– a database design methodology"
Proc COMPDEC conference,Los Angeles (1984)

[EICK 84b] Eick,Ch F
"Methoden und rechnergestutzte Werkzeuge fur den logischen Datenbankentwurf "
PhD-thesis university of Karlsuhe (July 1984)

[EIRA 85] Eick,Ch ,F , Raupp Th
"Decentralized database design using multityped functional and existence dependencies"
submitted for publication

[ELNA 84] Navathe,S Sashidbar T ,Elmasri R
"Relationship merging in schema integration"
Proc VLDB,p 78-90,
Singapore (1984)
Elmarsi,R ,Navathe,S
"Object integration in logical database design"
Proc COMPDEC conference, Los Angeles (1984)

[FEI 77] Feigenbaum,L
"The art of artificial intelligence
Themes and case studies in knowledge engineering"
Proc IJCAI Boston (1977)

[HOP 83] Hoeppner,W et all
"Beyond domain dependence"
Proc IJCAI,Karlsruhe (1983)

[GRO 83] Grover,Mark
"A pragmatic knowledge acquisition methodology"
Proc IJCAI,Karlsruhe (1983)

[MAP 83] Martin,P ,Appelt,D ,Pereira F
"Transportability and generality
in a natural language interface system"
Proc IJCAI, Karlsruhe (1983)

[McD 83] McDermott,John
"Extracting knowledge from Expert Systems"
Proc IJCAI,Karlsruhe (1983)

[McL 78] McLeod D
"A semantic data model and its associated structured user interface"
PhD thesis, MIT/LCS/TR214 (1978)

[MIT 82] Mitchel,T
"Generalization as search"
Artificial Intelligence p203-226 (1982)

[MO 82] Moto-oka
"Fifth generation computer systems"
North Holland (1982)

[NAGA 82] Navathe,S ,Gadgil,S
"A methodology for view integration in logical database design"
Proc eigth VLDB
conference,Mexico-City (1982)

[PIL 84] Pilote,Michel
"A framework for the design of linguistic user interfaces"
PhD-thesis, CSRG technical note #32,
University of Toronto (1984)

[RAU 84] Raupp,Th
"Qualitatsverbessernde Transformationen konzeptueller Schemata
– Grundlagen, Verfahren und Werkzeuge "
diploma thesis, university of Karlsruhe (August 1984)

[WED 80] Wedekind, H , Ortner, E
"Systematisches Konstruieren von Datenbankanwendungen"
Hanser Verlag (1980)

[WEI 81] Weiner,J
"Deriving database specification from user queries '
2nd Berkeley's Workshop on distributed Data Management and
Computer Networks (1981)