A NATURAL LANGUAGE APPROACH FOR REQUIREMENTS ENGINEERING

C. ROLLAND[1] C. PROIX[2]

ABSTRACT : The term Requirements Engineering refers to this part of a database development cycle that involves investigating the problems and requirements of the users community and developing a conceptual specification of the future system.
Natural language plays an important role during this stage that has proved to be crucial in the development of computerized systems. The required acquisition of application domain knowledge is achieved either through documents and texts analysis or by means of interviews i.e through language manipulation. Similarly validation of the specification is made via oral discussions with users.
The paper proposes that Requirements Engineering (R.E) should be supported by a CASE tool based on a linguistic approach. It presents a R.E support environment that generates the conceptual specification from a description of the problem space provided through natural language statements. Complementary, validation is based on texts generation from the conceptual specification to natural language. The paper focuses on the linguistic approach, demonstrates its generality and overviews its implementation in a CASE tool.

KEY WORDS : Requirements engineering, Natural language analysis, conceptual schema, information system design, text generation

1. Introduction

The need for modelling techniques by which systems may be described in high level conceptual terms has been recognized in the earlier phases of Databases and Information Systems (DB/IS) development in industry, business and administration.

---

[1] Université de Paris 1, 17 rue de la Sorbonne, 75231 Paris cedex 05, France
[2] Société CRIL, 146 Boulevard de Valmy 92707 Colombes cedex, France

This has caused the introduction of various conceptual models that have proved to be extremely useful to build in a high level specification of the future system (the so called conceptual schema) before this system is developed. (see the survey presented by Hull and King [Hull 87] for example).

However, the task of constructing the conceptual schema remains problematical. The route to reach the conceptual schema e.g the conceptual modelling process has the purpose of abstracting and conceptualizing the relevant part of the application domain. This is guided by requirements. The term Requirements Engineering introduced by Dubois [Dubois 89] has been used for this part of the DB/IS development that involves investigating the problems and requirements of the users community and developing a specification of the future system. The succeeding phase, where this specification is realized in a working system which is verified against the specification may be called Design Engineering [Bubenko 90]. Figure 1.1 shows the organization of DB/IS development cycle based upon requirements and system engineering.

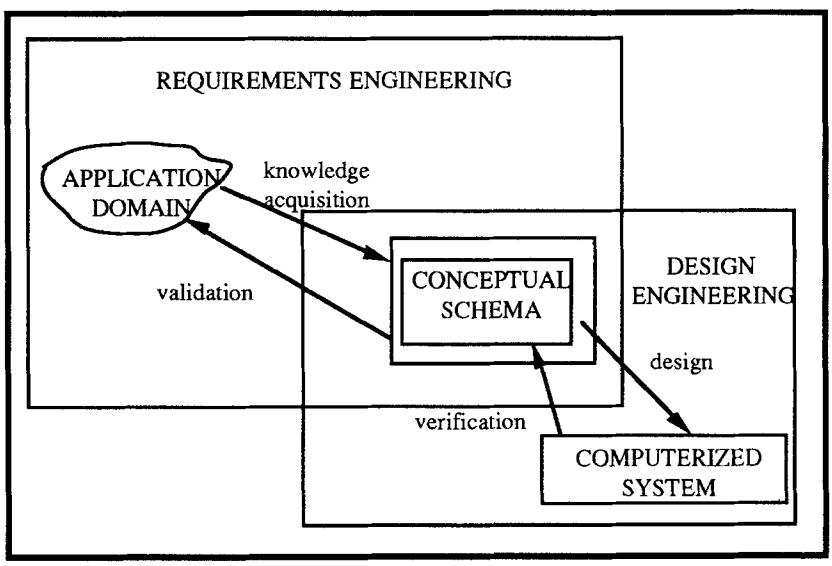

Figure 1.1 : DB/IS development cycle

Requirements Engineering consists of knowledge acquisition and validation.

The **acquisition task** falls into two areas, namely, analysis and modelling. The Requirements Engineering process starts with an observation of the real world, in order to identify pertinent real phenomena, their properties and constraints, and to classify similar phenomena into classes. Then the analyst represents and describes the classes, their properties and constraints through types of a specific conceptual model. Analysis leads to problem-statements, while modelling allows the description of elements of the conceptual schema.

The **validation task** has the objective of checking whether the conceptual schema is consistent and whether it correctly expresses the requirements informally stated by the users.

In many cases, analysts are able to correctly use concepts of a model but have difficulties to abstract reality in order to represent it through these concepts. This is similar to school students who are able to use simple equations but have many difficulties to build in equations from problem-statements. Similarly correcting a conceptual schema is easy while validating its adequacy to requirements is more difficult.

Analysis, modelling and validation are cognitive processes. However, analysis is based on domain-dependent knowledge, modelling requires model-dependent knowledge and validation requires both. More generally, Vitalari has shown, [Vitalari 83], [Vitalari 85], that experienced analysts use different categories of knowledge namely : organization specific knowledge, application domain knowledge, development methodology knowledge and functional domain knowledge.

It is the authors' belief that there is a need for CASE tools that support the Requirements Engineering process in a way that better reflects the problem solving behaviour of experienced analysts. This requires to identify, understand and formalize the cognitive mechanisms that allow the analyst to abstract reality and to represent it through concepts and to diagnose the specification from users points of view.

OICSI[1] (French acronym for intelligent tool for information system design) is a system prototype based on this premise. It exploits knowledge-based paradigms to provide an active aid to DB/IS analysts during the Requirements Engineering process. OICSI supports the analysts in the process of problem-statements acquisition, elicitation, modelling and validation.

In addition, the authors recognize that Requirements Engineering is mainly based on abstraction and have granted a privilege to a natural language approach.

Indeed, psychological research works dealing with the study of abstraction mechanisms show that abstraction is strongly interlocked with language manipulation.

Following this line, problem-statements in OICSI are expressed with the French natural language and automatically interpreted in terms of the OICSI conceptual model. Complementary, OICSI uses a text generation technique to feed back to the user information about the specification (i.e the conceptual schema).

This choice is enhanced by the fact that analysts do not proceed by direct observation of the real world but through a media which is the natural language. Indeed, the two most common ways for acquiring application domain knowledge are interviews and studies of existing documents (forms, legal documents...).

---

[1] OICSI is the name used in the academic area; in the industrial world this case tool is named ALECSI, it is developed by CRIL company.

According to the OICSI paradigm illustrated in figure 1.2, the analysis task refers to the description of the relevant real world phenomena using the French natural language, the modelling task refers to the mapping of problem-statements onto basic concepts of the OICSI underlying DB/IS development methodology and the validation task is based upon a paraphrased description of the conceptual schema in the French natural language.

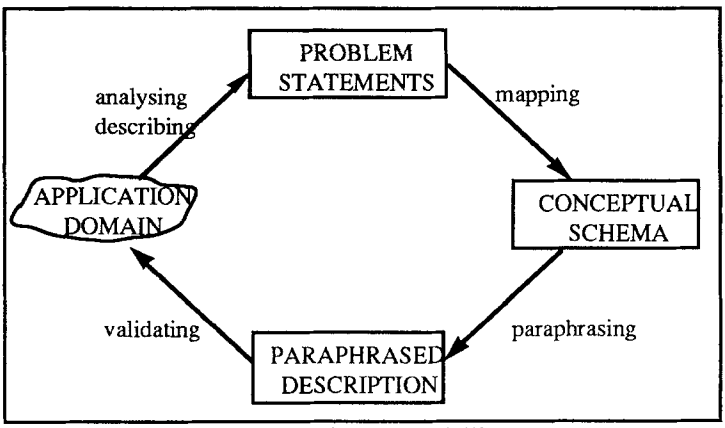

Figure 1.2 : Analysis and Modelling process.

Using OICSI, the conceptual schema is hidden to the future system users. The "system referential", they have to understand, comment upon and validate, is expressed using natural language. Even for the analysts the conceptual model and the conceptual schema are partially hidden since OICSI automatically supports modelling as well as text generation.

However, it must be mentioned that OICSI provides a graphical interface. Depending on their personal abilities to understand conceptual modelling, the analysts will use the most appropriate interface.

Similar approaches to solving Requirements Engineering from the description of the application domain uttered with natural language sentences have been followed for example in the AMADEUS project which aims at combining graphics and natural language [Black 87]. Others examples are SECSI [Bouzegoub 86] and ACME [Kersten 86] which are conceptual modelling expert systems.

Text generation has been used in different areas of databases : it is an important matter in natural language interfaces to databases; it also used for tutorial purposes in learning a query language and for generating readable error messages. Examples of prototype systems are EXPOUND [Chester 76] which translates formal proofs into English, CO-OP [McKeown 86] based upon a syntactic approach, PERFORM [Muckstein 85] and ELFS [Luk 86] which are knowledge based approaches for text generation from SQL to natural language, De Roeck paraphrasing of relational calculus [De Roeck 88] and Grishman paraphrasing of predicate logic [Grishman 79].
The remainder of this paper describes the natural language approach for Requirements Engineering and its implementation in OICSI. Section 2 presents the linguistic approach for requirements acquisition and elicitation. The paraphrasing mechanism for validation is presented in section 3. A brief overview of implementation aspects is given in section 4.

2. The linguistic approach

Conceptual modelling in OICSI is based on a linguistic approach that tries to formalize the linguistic mechanisms through which analysts are able to abstract observed phenomena onto concepts.

The problem-solving behaviour of analysts is first intuitively introduced. The "CASE for CASE" theory (which is the foundation of the formalization of the analyst behaviour) is thus recalled. Finally, our linguistic approach is detailed and the conceptual schema generation is presented.

2.1 Intuitive introduction to analysts problem solving behaviour

This section is an attempt to highlight the linguistic mechanisms used by analysts.

Let us imagine that our favourite analyst *Ado* is used to manipulate the Entity-Relationship (E-R) model [Chen 76]. This means that *Ado* will try, when observing the real world, to identify classes of real world phenomena that can be modelled as entity types, attributes or relationship types.

Thus, during an interview, if *Ado* hears the sentence:

"A subscriber has a name and an address."

He will probably introduces in the conceptual schema an entity type SUBSCRIBER with two attributes NAME and ADDRESS.
Now, in order to understand the analyst behaviour, let us ask the question :"How did *Ado* get this result?".

A first response could be that *Ado* knows the meaning of the words "*subscriber*", "*name*" and "*address*", and how they relate one with others. This means that *Ado* uses a kind of common-sense knowledge to match the sentence onto the E-R schema. This knowledge is based on couples (word, real object) which allow to relate a word to a well known object in the real world.

But assume now that the sentence is :

"The colydrena have a pedistylus and a folicul."

As *Ado* did, many analysts will make the hypothesis that the word "*colydrena*" is a non lexical object type that can be modelled by an entity type and that "*pedistylus*" and "*folicul*" are two attributes related to the entity type. *Ado* is not certain that he did the right interpretation of the sentence but the interpretation is plausible and he can, later, validate its truth discussing with domain specialists.

In this case, *Ado* did not use the same kind of common-sense knowledge as previously. He does not know the meaning of the words (they are imaginary), but, however without any understanding of the words he found a model of the described situation (which is, indeed, correct).

Ado's reasoning is based on the recognition of a particular sentence pattern which is colloquial to him. The knowledge which is used, is a linguistic knowledge related to language manipulation. It allows him to recognize and to interpret the following sentence pattern :

<Subject Group><Verb expressing ownership><Complement Group>

The pre-established interpretation of such pattern allows Ado to associate the subject group of the sentence to a real entity class as the owner of the attributes represented by the complement group's words.

The linguistic knowledge is certainly the most common knowledge within the analysts population. Analysts use it, sometimes explicitly, but most often in an implicit way. Our goal is to make explicit the different types of sentence patterns in order to formalize this kind of linguistic knowledge and to support the process of the problem-statements interpretation and modelling in a computerized way.

The linguistic approach implemented in OICSI is borrowed from the Fillmore's theory "Case for Case" [Fillmore 68].

Section 2.2 summarizes the main points of this theory. Its specialization for OICSI is presented in section 2.3.

2.2 The Fillmore's case system

The main concept of the Fillmore's theory is the notion of case introduced as follows: "*the case notions comprise a set of universal, presumably innate, concepts which identify certain types of judgement human beings are capable of making about the events which are going on around them...*".

Cases are types of relationships that groups of words have with the verb in any clause of a sentence. One of the basic Fillmore's assumption is that it exists a limited number of cases. Fillmore exhibits six major cases: AGENTIVE, INSTRUMENTAL, DATIVE, FACTITIVE, LOCATIVE and OBJECTIVE.

(1) John opens the door.
(2) The door is opened by John.
(3) The key opens the door.
(4) John opens the door by means of the key.
(5) John uses the key in order to open the door.
(6) John believes that he will win.
(7) John is ill.

Figure 2.1 : Examples of sentences

For example in sentences (1) and (2) of the figure 2.1 "*John*" is associated to the case AGENTIVE and "*door*" to the case OBJECTIVE; the word "*key*" in sentences (3), (4), (5) is associated to the INSTRUMENTAL case, while in sentences (6) and (7) "*John*" is associated to the DATIVE case.

Obviously, the same word can correspond to different cases in different sentences.

One complementary assumption of the Fillmore's theory is that the meaning of any clause is derivable from the meaning of the verb and the recognition of embedded cases. This leads to the identification of predefined patterns with associated derivable meanings.

For example, due to the fact that sentence (1) has a structure of the type:

<Verb expressing action, AGENTIVE, OBJECTIVE>

allows to infer that "*John*" is the agent who performs the action on the object "*door*".

Sentences (1) and (2) correspond to the previously mentioned structure; the structure of sentence (3) matches the type :

<Verb expressing action, INSTRUMENTAL, OBJECTIVE>

and finally, sentences (4) and (5) have the following pattern :

<Verb expressing action,OBJECTIVE, AGENTIVE, INSTRUMENTAL>.

The Fillmore's patterns allow to perform a classification of natural language sentences with regards to their structure and, thus, to infer their meaning according to the class they belong to.

2.3 Specialization of the Fillmore's case system

Experimentations of the Fillmore's theory convinced the authors that the theory was applicable and pertinent to support the DB/IS analysis and modelling process. However, we reach the conclusion that the cases might be adapted to the purpose of establishing problem-statements allowing the construction of an DB/IS conceptual schema. Indeed statements about real world phenomena fall into two categories: fact descriptions and rules.

Examples of fact descriptions (we consider a subscription library system) are as follows:

(1) In the library, a book is described by a unique reference number, the authors' names, the publisher name and the year and version of editing.
(2) Last and first names of the subscriber, his address, first year of subscription and last date of subscription fees payment are recorded.
(3) The status of each copy of a book is recorded in real time.

Our understanding of facts is similar to the Nijssen's approach [Nijssen 89].

The following are examples of rules:

(1) Subscription fees are paid every year.
(2) A subscriber, properly registered (i.e who paid the fees) is called an "active" subscriber.
(3) A subscriber cannot borrow more than three books at the same time.
(4) Books are only loaned to active subscribers.
(5) When a loan request cannot be satisfied it becomes a "waiting request".
(6) After 13 months without paying the subscription fees, the subscriber status becomes "inactive".
(7) "Waiting request" are treated in their chronological order.

As just exemplified, rules can express management rules independent or dependent of time, static constraint rules or dynamic constraint rules .

Sentences describing either facts or rules are the problem-statements that OICSI automatically interprets by performing a case approach.

2.3.1 The case classification

The case notion has been extended in two directions: cases are applicable to clauses and the classification of cases has been revised.

. According to the Fillmore's theory, cases relate to words in sentences. It is the authors' belief that the notion of case could be successfully applied not only to words but also to clauses in sentences. This allows to interpret a complex sentence in a top-down fashion. The case approach is first applied to subordinate clauses with regards to the verb of the main clause. Thus, the case approach is again applied to each of the subordinate clause.

. The classification of cases used by OICSI is as follows :
<OWNER, OWNED, ACTOR, TARGET, CONSTRAINED*, CONSTRAINT*, LOCALIZATION* , ACTION* , OBJECT>.

We exemplified the meaning of these cases on the following set of sentences.

> (1) A subscriber is described by a name, an address and a number.
> (2) A subscriber borrows books.
> (3)When a subscriber makes a request of loan, the request is accepted, if a copy of the requested book is available, else the request is delayed.

In sentence (1), "*subscriber*" is associated to the OWNER case and "*name*", "*address*" and "*number*" are associated to the OWNED case.

In sentence (2), "*subscriber*" is associated to the ACTOR case and the OWNER case, while "*books*" is associated to the OWNED case; these two cases express that there is a relationship between "*subscriber*" and "*books*". The entire clause is associated to the ACTION case.

In sentence (3) :

- the clause "*When a subscriber requests for a loan*" is associated to the LOCALIZATION case,
- inside this clause, the phrase "*request of loan*" is associated to OBJECT case,
- the clause "*if a copy of the requested book is available*" is associated to the CONSTRAINT case,
- the clause "*the request is accepted*" is associated to the ACTION and the CONSTRAINED case,
- inside this clause, the word "*request*" is associated to the TARGET case.

---

* denotes cases that may be applied to clauses

Complementary, classes of verbs have been identified. The figure 2.2 shows both the hierarchy of classes and some examples of class instances.

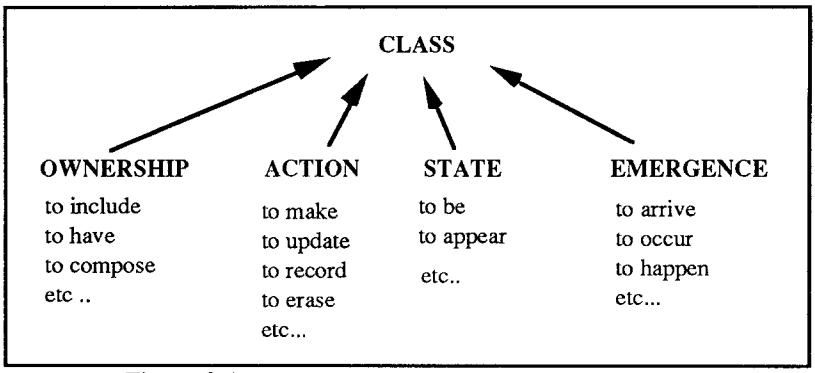

Figure 2.4 : Hierarchy and instances of classes of verbs

2.3.2 The linguistic patterns

A set of patterns that combine cases and classes of verbs previously introduced have been defined. These patterns are of two different types:

- elementary patterns allow to associate cases to syntactic units of a clause,
- sentence patterns allow to associate cases to clauses of a sentence.

Both are introduced and exemplified in turn.

Elementary patterns

They fall again into three different categories:

- structural pattern,
- behavioural pattern,
- constraint pattern.

SP1 and SP2 are examples of simple **structural patterns.**

SP1 : [Ng_subject](OWNER) [verbal form](ownership_subject)
[Ng_complement](OWNED)

SP2 : [Ng_subject](OWNED) [verbal form](ownership_complement)
[Ng_complement](OWNER).

The notation [syntactic unit](case) means that the "syntactic unit" is associated to the case "case". The following abbreviations Ng, Cl, Sub, Mn, are respectively used to refer to a Nominal group, a Clause, a Subordinate clause and a Main clause.

The clause : "*any subscriber has a name and an address*" matches the SP1 pattern and can be interpreted in the following way:

- the clause subject "*any subscriber*" plays the role of OWNER,
- "*has*" is the verb belonging to the ownership class,
- "*a name* " and "*an address*" are subject complements playing the role of OWNED.

It is obvious that patterns of the SP1 family are appropriated to fact sentences.

The sentence " *loan-requests are made by subscribers*" can be unified to pattern SP2.

BP1, BP2, BP3, and BP4 are four examples of **behavioural patterns.**

BP1 : [Ng_subject](ACTOR) [verbal form](action)
[Ng_complement](TARGET)

BP2 : [Conjunction](LOCALIZATION) [Ng_subject](ACTOR)
[verbal form](action) [Ng_complement](OBJECT)

BP3 : [preposition](LOCALIZATION) [Ng](OBJECT)

BP4 : [Ng](TARGET) [verbal form](action)

"*Subscribers borrow books*" is a clause that matches the BP1 pattern :
- "*subscribers*" as the subject of the clause plays the role of ACTOR,
- "*borrow*" is a verb belonging to the action class,
- "*books*" is the subject complement which plays the role of TARGET.

The clause : "*when a subscriber returns a book copy*" can be unified with BP2 pattern with the following interpretation:
- "*when*" is a conjunction that expresses the LOCALIZATION of the action,
- "*a subscriber*" is the subject that plays the role of ACTOR,
- "*returns*" is the verb that belongs to the action class,
- "*a book copy*" is the complement that plays the role of OBJECT of action.

BP3 is a pattern which deals with circumstantial complements and, for this reason, is not organized around the verb but around the preposition.

Within the clause: "*As soon as the receipt of a subscriber's subscription fees, the subscriber's status is updated*", the phrase "*As soon as the receipt of a subscriber's subscription fees*" matches the BP3 pattern with the following interpretation:
- "*As soon as*" is the preposition that describes the LOCALIZATION of action expresses by the clause,
- "*the receipt of a subscriber's subscription fees*" is the phrase that plays the role of OBJECT.

Finally the BP4 pattern allows to interpret a particular type of clauses which describe actions such as "*the loan is agreed upon*".

At last CP1 is an example of **constraint pattern.**

CP1 : [Ng_subject](CONSTRAINED) [verbal form](state)
[Ng_complement](CONSTRAINT)

The clause: "*the number of loans is equal or less than three*", can be unified to the CP1 pattern in such a way that:

- "*the number of loans*" plays the role of CONSTRAINED, and
- "*equal or less than three*" is the predicate group associated to the CONSTRAINT case.

Sentence patterns

The sentence patterns define the cases of embedded clauses in a same sentence. They are constructed combining elementary patterns. Let us consider two examples:

SPT1 : [Main clause]

SPT2 : [Subordinate clause unifying a BP pattern](LOCALIZATION)
[Subordinate clause unifying a BP2 pattern](CONSTRAINT)
[main clause unifying a BP pattern with a verb expressing an action](ACTION + CONSTRAINED)

SPT1 corresponds to sentences composed with only one main clause. This clause must be able to match :

- either a structural pattern; the sentence "*A subscriber is described by his name and his address*" is an example of it,
- or a behavioural pattern with a verb expressing an action; "*Subscribers borrow copies of books*" matches this pattern. The ACTION case is thus affected to the sentence,
- or a constraint pattern; this corresponds to the sentence "*The number of loans is limited to three*". This sentence is associated to the CONSTRAINT case.

The subordinate clause that can be unified to a behavioural pattern determines the spatio-temporal LOCALIZATION of the action described by the main clause.

The sentence: "*When there is a loan request, the loan is agreed only if the subscriber's status is "active" and if a copy of the requested book is available*" corresponds to the SPT2 pattern :

- the clause "*When there is a loan request*" matches the BP2 pattern and is associated to the LOCALIZATION case;
- the clauses "*only if the subscriber's status*" and "*if a copy of the requested book is available*" match the CP1 pattern and are associated to the CONSTRAINT case.
- the clause "*the loan is agreed*" matches the BP4 pattern and corresponds simultaneously to the ACTION and CONSTRAINED cases.

2.4 Conceptual schema generation

We assume that it is possible to simply link cases and concepts. Thus the conceptual schema generation is grounded upon rules that map cases onto concepts. These rules are dependant of the target conceptual model. Conversely the linguistic patterns are independent of a particular modelling technique and can be used within any design methodology.

Figure 2.3 gives a brief overview of the main mapping rules implemented in the OICSI envirronment. We recall that OICSI is based upon the REMORA methodology [Rolland 82] which identifies four basic concepts namely, objects, actions, events and constraints. A detailed description of this aspect can be found in [Rolland 87]. These are the four type of nodes of the semantic net used by OICSI to implement the conceptual schema under construction. Arcs of the net are of five types :

- rl : expresses a relationship between two objects nodes;
- md : expresses that an action modifies an object;
- tr : expresses that an event triggers an action;
- act : expresses that an object has a particular state change which is an event;
- ct : connect a constraint to the node (object, action or event) which is constrained.

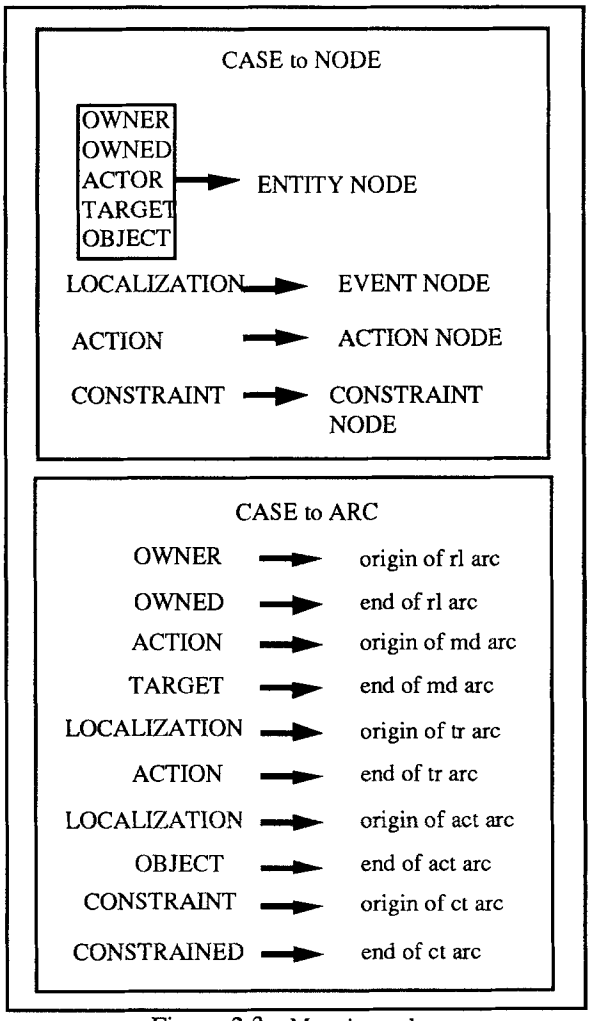

Figure 2.3 : Mapping rules

3. Conceptual schema validation and paraphrasing

The Requirements Engineering process includes also the validation cycle. In order to base the whole Requirements Engineering process on a natural language approach, we propose to feed back to the user information about the conceptual schema using again the French natural language.

The paraphrasing technique we have developed has the scope to generate natural language texts using the words and expressions of the users community and avoiding to describe the conceptual schema contents in technical terms.

We introduce first the main principles of the techniques used for text generation and then we present our solution to conceptual schema validation by paraphrasing.

3.1 Principle of natural language generation

A system for text generation must be able to select information from some knowledge base and to organize it into a natural language text. Several approaches have been proposed for this purpose. Most of them use the distinction between the "what to say" from the "how to say". However, they differ from the degree of overlap of these two aspects.

The "what to say" deals with the determination of informations which are relevant for the purpose of the text, with what the users need to know, and how much detailed an object or event must be described.

The "how to say" deals with the choice of a linear order for the information selected, specifying how to aggregate the information (determination top-form paragraphs and sentence boundaries).

The structuralist approach, which is mainly represented by Bloomfield [Mounin 72], [Harris 85], admits this distinction but concentrates the semantics in the "how to say".

The fonctionnalist approach [Harrys 85] is not aware of this distinction. In this approach the "what to say" and the "how to say" are mixed.The sentences are directly built from the knowledge base.

Finally the third approach admits that the major part of the semantic is included in the "what to say" and the minor part of it is in the "how to say" [Chomsky 57]. Chomsky who has initially followed the structuralist approach is the father of this third approach.

Among the set of possible solutions we have retained the Chomsky approach [Chomsky 65] .
The basic Chomsky assumption is the existence of a underlying structure, namely the deep structure, to any sentence in any human language. In addition, there is an infinite number of ways, namely the surface structures to represent the deep structure in different languages.

The deep structure expresses the semantics of a sentence by means of semantic elements and relationships among them. It corresponds to the "what to say".

Grouped all together, the deep structures corresponding to a knowledge base, allow us to reach a semantic understanding of its contents.

The surface structure represents each sentence of a text by means of a set of phrases. It corresponds to the "how to say". Many sets of surface structures may correspond to the same deep structure. In addition, it is possible to define a set of transformation rules (a generative grammar [Chomsky 69]) which allow to map a deep structure into an infinite set of surface structure.

Based upon this distinction the process of generating natural language texts is summarized in figure 3.1.

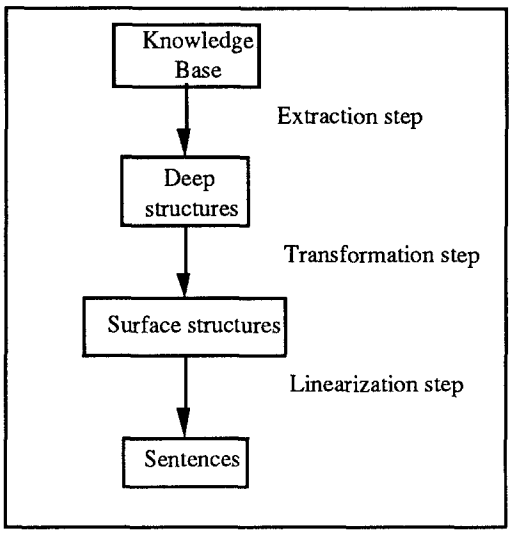

Figure 3.1 : process of generating natural language texts

It is assumed that the knowledge base provides the description of some application domain.

The first step consists of defining the appropriate deep structures for the knowledge base contents. Deep structures are often represented through semantic nets.

The second step maps the deep structure onto a surface structure. This step uses a generative grammar [Chomsky 69] which allows to produce skeletons of sentences in the target natural language. This surface structure includes all the phrases of the future sentence and its grammatical structure.

The last step, so called linearization step, uses the surface structure to produce a readable sentence. This step uses a lexical knowledge base in order to solve problem such as :

- determination of valid articles,
- tacking into account singular, plural, ...
- use of idiomatic forms,
- phonological short-cuts.

It is eventually possible to complete the process by a structuration step which aims at reorganizing the collection of sentences into chapters, sections and paragraphs.

3.2 The OICSI paraphrasing process

Following the Chomsky's guidelines we have organized the process for paraphrasing from the conceptual schema to a French text into a similar way which is shown in figure 3.1.

The knowledge base mentioned in figure 3.1 is the OICSI base of facts i.e. the semantic net which represents the conceptual schema under construction.

The deep structure definition consists of grouping nodes and arcs of the semantic net. As a matter of fact, two rules are used in order to group in a same deep structure :

- all the nodes and arcs describing an entity,
- all the nodes and arcs describing an event and its triggered operations.

We name a situation of the semantic net, a set of nodes and arcs which correspond to a deep structure.

This solution is motivated by the fact that we want to restitute to the users descriptions of their application domain as close as possible to the problem statements they have initially provided to the system. Following our assumption in section 2.3 , we consider that facts and management rules are the two easier entry points for users in the process of developing information systems.

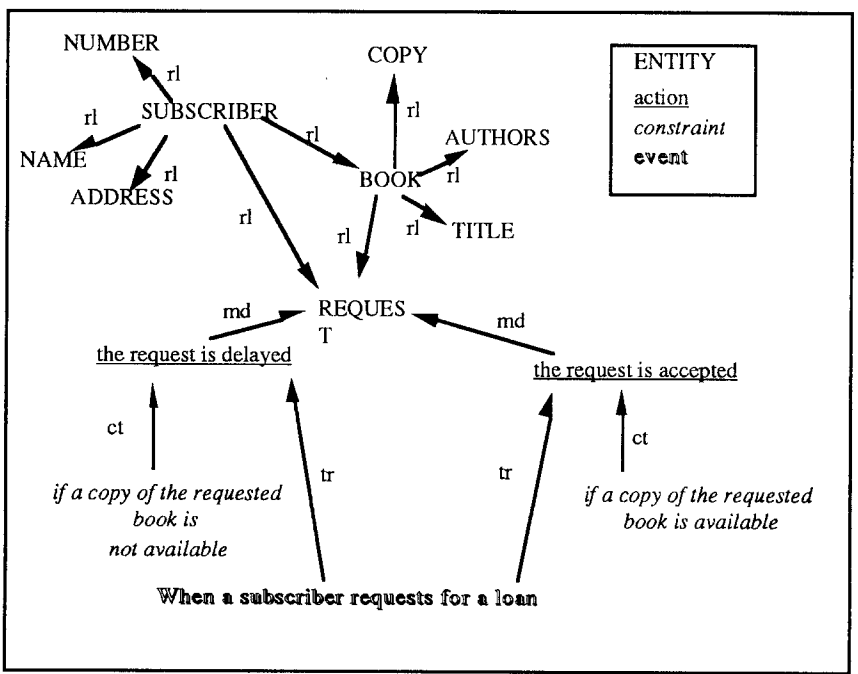

Figure 3.2 : The graphical representation of the conceptual schema

Thus the text generated by the system will describe :
- on one hand, the static aspects of the world through entities, their properties and relationships;
- on the other hand, the behavioural aspects through rules with the standard pattern "*when event, if condition then action*".

For example, from the conceptual schema presented in the figure 3.2, the system recognizes the two deep structures shown in the figure 3.3.

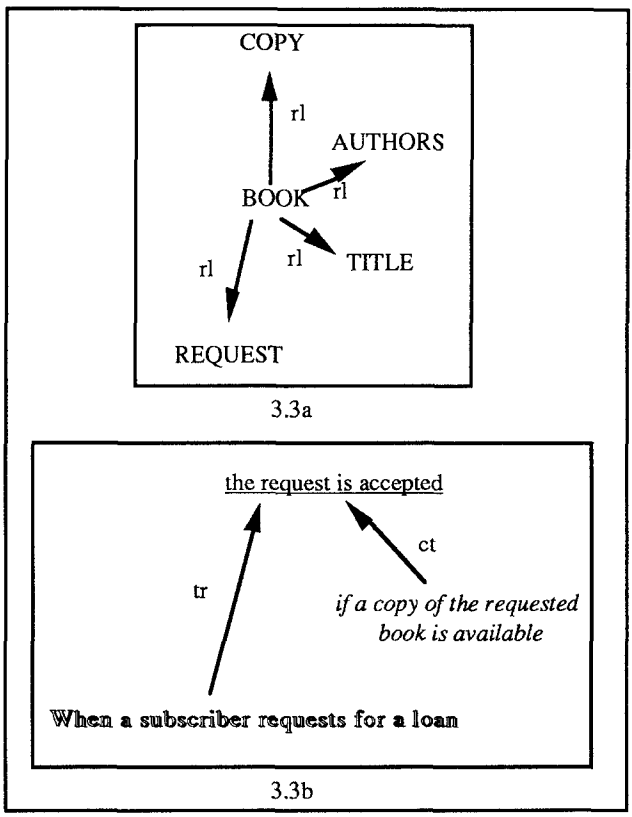

Figure 3.3 : The two deep structures recognized from figure 3.2

The 3.3a deep structure includes all informations about the entity BOOK, and the 3.3b deep structure includes all informations about the event "*a subscriber requests for a loan*".

The second step is the transformation of a deep structure into a surface structure. We make the hypothesis that the number of different types of situations in the conceptual schema is limited and that these situations are well defined. Therefore, the transformation step is based on a set of surface structure patterns which are associated to each type of situation.

For example, the 3.3a deep structure will be associated to the following surface structure:

sentence(**verb**(*to have*), **subject**(*book*), **complement**(*title, many copies, one or many authors, one or many requests*)).

The 3.3b deep structure corresponds to the following surface structure :

sentence(**circumstantial proposition**(*a subscriber requests for a loan*),
conditional proposition (*if a copy of the requested book is available*),
main proposition(*the request is accepted*)).

The last step is the linearization phase. Using lexical knowledge and the surface structure this step produces readable sentences. The main tasks realized here is :

- to conjugate correctly the verbs of the sentence;
- to determine the conjunctions for the subordinate propositions;
- to select the appropriate articles.

For example, the sentences produced from the previously defined surface structure are :

- A book has a title, many copies and one or many authors.
- When a subscriber requests for a loan, if a copy of the requested book is available then the request is accepted.

4. Implementation overview

The two processes, namely the conceptual schema generation process and the paraphrasing process are implemented in an expert system approach. This means that the two processes are performed by an inference engine which uses rules. For modularity and flexibility reasons the rules are Prolog production rules.

We limit ourselves to a brief overview of the two processes mentioning the different classes of rules and their role.

4.1 The conceptual schema generation process

The OICSI inference engine uses three main classes of rules :

- lexical and syntactic rules,
- linguistic rules,
- mapping rules,

in order to progressively transform NL sentences onto nodes and arcs of the semantic net. The process is organized into three steps.

. During the **analysis step** the system builds an internal representation of the initial sentences by means of syntactic trees, with the purpose of decomposing each sentence into grammatical unit.
This part of the process is based on wellknown techniques developed for the general purpose of natural language recognition [Bruce 75], [Cordier 79] and [Kayser 81].

The role of lexical rules is to determine the grammatical nature of each word of any clause of a sentence and to classify the verb clause into the four classes: ownership, action, state, emergence. Lexical rules use a dictionary which contains information about the grammatical nature of words and about the meaning and the classification of verbs. Syntactic rules allow the system, on one hand, to verify that a sentence belongs to the authorized language, and, on the other hand, to build up the syntactic trees. These rules are based on the use of a generative grammar which corresponds to the system's grammatical knowledge.

. During the **linguistic step**, the system makes pattern matching in order to unify each syntactic tree with one of the sentence pattern defined in section 2, and to associate each syntactic unit with a case. Pattern matching and association of cases to the phrases of a sentence is performed simultaneously in the same rule. Basically any linguistic rule as the following form :

- the premise of the rule correspond to the conditions that allow to recognize the sentence (or clause) pattern,
- the conclusions of the rule associate cases to elements of the sentence (or clause).

Patterns recognition is based both on the class of the verb (as identified during step 1 and attached to it in the syntactic tree) and on the grammatical structure of the sentence (or clause). Generally, a pattern is implemented through a set of linguistic rules in order to take into account the variety of grammatical structures. As an illustration, rules RL1 and RL2 are two examples of rules necessary for implementing the pattern SP1.

RL1 :
IF meaning(clause(verbal form)) = ownership_subject
AND gram_structure(Ng_subject) = <article, noun_1>
AND gram_structure(Ng_complement) = <article, noun_2>
THEN case(noun_1) = OWNER
case(noun_2) = OWNED.

RL2 :
IF meaning(clause(verbal form)) = ownership_subject
AND gram_structure(Ng_subject) = <article, noun_1, predicate_1>
AND gram_structure(Ng_complement) = <article, noun_2>
AND gram_structure(predicate_1) = <preposition, article, noun_3>
THEN case(noun_1) = OWNER(verb)* and OWNED(predicate)
case(noun_2) = OWNED.

* the notation OWNER(verb) and OWNED(predicate) mean that the role OWNER is played in regards to the verb and that the role OWNED is played in regards to the predicate. By default, the case meaning is in regards to the verb.

. Finally, the **mapping step** consists of building the semantic net. Each syntactic tree is mapped onto a set of nodes and arcs of the semantic net. Mapping rules implement the relationship summarized in figure 2.3 (see section 2). They allow to automatically build nodes and arcs of the semantic net from cases and patterns determined in the previous step.

4.2 The paraphrasing process

Similarly the OICSI inference engine uses three main classes of rules :

- extraction rules,
- transformation rules,
- linearization rules,

in order to perform the three steps of the paraphrasing process illustrated in figure 3.2.

. **Extraction rules** are used to cluster nodes and arcs related to either an entity or an event type and to construct the corresponding deep structure.

. **Transformation rules** allow to map the deep structures into surface structures. A pattern matching mechanism is used in order to associate to a deep structure the appropriate surface structure.

. **Linearization rules** are used in order to rewrite a surface structure into a readable sentence. They include rules for to conjugate the verbs, to select the article and so on. A major part of these rules use a dictionary which represents the lexical knowledge of the tool.

Obviously, the two processes (conceptual schema generation and paraphrasing) are performed in an interactive way. For example the user's aid may be solicitated during the analysis step to add a new verb in the dictionary. At any time the user can ask for explanation about the system deductions and this can lead to pattern transformation. At last the analyst/user is allowed to directly manipulate the semantic net through a graphical interface in order to add, delete or change any arc or node of the net. In addition, the two processes are fully integrated. This means that the user can ask for paraphrasing from the conceptual schema at any point of its generation process. This allow to constantly keep the equivalence between a set of natural sentences and the formalized conceptual schema. We believe that this is helpful to validate the conformance of the system specifications to the user requirements.

Similar considerations have been discussed as the premise of the RUBRIC [Van Assche 88] and TEMPORA [Loucopoulos 90] projects.

A more detailed description may be found in [Loucopoulos 92].

5. Conclusion

The paper has argued that the natural language plays an important role during the DB/IS development cycle. Therefore, the ideas that Requirements Engineering should be supported by a Case tool based on a linguistic approach and that validation of specifications must be performed by means of text generation technique have been presented.

In a first time, the work reported in this paper is based on the premise that Requirements engineering is strongly interrelated to language manipulation. It represents an attempt at improving problem-statements elicitation, interpretation and modelling through the use of a linguistic approach. It is proposed that the problem-statements for an information system development should be expressed via natural language sentences.

The work reported presents how a linguistic approach based on the Case notion can be used to automatically carry out the IS modelling. The paper details the linguistic approach and its implementation in the expert design system, known as OICSI. The thesis put forward in the paper is that the linguistic approach is general, in the double sense that it can be customized for different modelling techniques and, in addition, it can be applied in a wider sphere of problems. From this point of view the work reported relates to other research works such KOD [Vogel 88] or SECSI [Bouzeghoub86].

In a second time, the paper presents some solutions based on theorical linguistic works in order to validate the conceptual schema by paraphrasing from conceptual schema to natural language texts. This paraphrasing technique has the scope to generate natural language texts with words and expressions of the users community and avoiding description of the conceptual schema contents in technical terms.

References

[Bouzeghoub 86] M. Bouzeghoub and G. Gardarin : "SECSI : an expert system approach for data base design", in Proc. of IFIP world congress, Dublin, Sept 1986.

[Bruce 75] B. Bruce : "Case systems for natural language", Artificial Intelligence Nb 6, 1975.

[Black 87] WJ. Black: "Acquisition of Conceptual data models from natural language descriptions, 3rd Conf. of the European chapter of ACM, Danemark, 1987.

[Bubenko 90] J. Bubenko et all : Syslab/Decode research plan Syslab report 1990.

[Chen 76] P.P.S Chen : "The entity relationship model : toward a unified view" ACM Trans. on data base systems, Vol 1, Nb1, 1976.

[Chester 76] D. Chester : "The translation of formal proofs into English", Artificial Intelligence, vol 7, n°2, 1976.

[Chomsky 57] N. Chomsky : "Syntactic strutures", Mouton Ed, The Hague 1957.

[Chomsky 65] N. Chomsky : "Aspects of the theory of syntax", MIT Press Ed, Cambridge Mass, 1965.

[Chomsky 69] N. Chomsky : "Language and Mind", Payot ed, 1969.

[Cordier 79] M. Cordier: Connaissances sémantiques et pragmatiques en compréhension du langage naturel, 2Çme congrés AFCET-INRIA, Reconnaissances des formes et Intelligence Artificielle, Toulouse 1979.

[De Roeck 88] A.N.D Roeck, B.G.T. Lowden : "Generating English paraphrases from formal relational calculus expressions" Coling (Pub) 1988.

[Dubois 89] E. Dubois, J. Hagelstein, A. Rifaut : "Formal requirements engineering with ERAE", Philips journal of research, vol 43, N) 3/4 1989.

[Grishman 79] R. Grishman : "Response generation in question answering systems" in ACL 1979.

[Fillmore 68] CJ. Fillmore : "The Case for Case", in Universals in linguistics theory; Holt, Rinehart and Winston, Inc., E. Bach/R.T. Harms (eds) 1968.

[Harris 85] M. Dee Harris : "Introduction to Natural Language processing", Reston Publishing company, 1985.

[Hull 87] R. Hull and R. King : Semantic Database Modeling : Survey, Applications and Research issues", ACM computing Surveys, vol 19, n-3, 1987.

[Kayser 81] D. Kayser : "Les ATN sÇmantiques" 3Çme congräs AFCET-INRIA, Reconnaissances des formes et Intelligence Artificielle, 1981

[Kersten 86] M.L. Kersten, H. Weigand, F. Dignum, J; Proom: "A conceptual modelling expert system", 5th Int. Conf. ont the ER Approach S. Spaccapietra(ed), Dijon, 1986.

[Loucopoulos 90] P. Loucopoulos et all : "From software engineering to business engineering: Esprit projects in information systems engineering", in CAISE'90, Int. Conference on : "Advanced Information System Engineering ", Springer-Verlag, 1990.

[Loucopoulos 92] "Conceptual modelling databases and Case: an integrated view of information systems development", P. Loucopoulos (ed), Mac Grawhill (Pub) 1992 (to be published).
[Luk 86] W.S Luk, S. Kloster : "ELFS: English language from SQL", ACM Trans. on Databases systems, vol 11, n°4, 1986.
[Mc Keown 86] K. Mc Keown : "Paraphrasing questions using given and new information", Am. journal of computational linguistics, vol 9 n°1, 1986.
[Muckstein 87] E.M Muckstein, M.G. Datovsky :" Semantic interpretation of a database query language", Data and Knowledge engineering, vol 1, 1985.
[Maddison 83] R. Maddison : "Information System methodologies", Wiley-Heyden 1983.
[Mounin 72] G. Mounin : "La linguistique du 20ième siècle", Presses Universitaires de France Ed, 1972.
[Nijssen 89] G.M. Nijssen, T.A. Halpin : "Conceptual Schema and relational database design : a fact oriented approach", Prentice-Hall, Englewood Cliffs, New Jersey, 1989.
[Olle 82] T.W. Olle, H.G. Sol and A.A Verrijn Stuart :"Information System design methodologies : a comparative review", (IFIP WG 8.1 CRIS 1) North Holland, Amsterdam , NL, 1982.
[Rolland 82] C. Rolland and C. Richard : "The Remora methodology for information systems design and management" in [Oll 82].
[Rolland 87] C. Rolland, G. Benci and O. Foucault : "Conception des systèmes d'information : la méthode REMORA", Eyrolles (Pub) 1987.
[Van Assche 88] F. Van Assche, P.J. Layzell, P. Loucopoulos and G. Speltinex : "Information Systems development : a rule based approach", in Journal of knowledge based systems, 1988.
[Vit alari 83] N.P. Vitalari and G.W. Dickson : "Problem solving for effective systems analysis : an experimental exploration ", in Comm. ACM Vol 26 N°11, (November 1983).
[Vitalari 85] N.P. Vitalari : "Knowledge as a basis for expertise in systems analysis : an empirical study", MIS Q, (September 1985).
[Vogel 88] : C. Vogel : "Génie cognitif", Masson collection Sciences cognitives, 1988.