Natural language discourse generation in a support tool for conceptual modeling

Hercules Dalianis

SYSLAB
Department of Computer and Systems Sciences
The Royal Institute of Technology and
Stockholm University
Electrum 230
S-164 40 Kista
SWEDEN
ph. (+46) 8 16 16 79
E-mail: hercules@dsv.su.se

Abstract. The support systems for conceptual modeling of today lack natural language feedback. The paper argues for the need of natural language discourse for the validation of a conceptual model. Based on this conclusion, a suggestion is made on a natural language discourse generation system as a validation tool and also as a support tool in executing a conceptual model. Various appropriate natural language discourses are then proposed in the paper. The proposed texts were analyzed with Hobbs' coherence relations and the discourses could be analysed with fewer coherence relations than the whole theory contains.

A discourse grammar is generalized from the analyzed discourses. A suggestion to control the generation is the user's question together with a user model and various feature checking of the grammar.

To conclude the paper a support system based on the natural language generation techniques of today and on previous working systems constructed by the author is suggested.

1. Introduction

During the construction of a large scale computer system one should decide on the functionality of the system, the so-called requirements engineering process, before starting its implementation. Therefore a formalism for modeling an information system has been developed, the so-called conceptual modeling language. A conceptual model must be validated to discover if it corresponds to the described domain. Since the conceptual model is described in a formal language the information can be difficult to understand and validate for an inexperienced user. This implies that natural language, (NL), is appropriate to use in the validation process. Another reason for using NL is that a novice user does not have to learn a complicated language or formalism to understand the conceptual model.

Natural language generation, (NLG), will give the end users a direct feedback of the semantics of the formal representation, we will in fact lower the conceptual barrier of the end users by using NL.

It is important to generate natural language discourses from a conceptual schema. The reason for the need of discourses, i.e. a set of logical interconnected sentences, and not single sentences, arouse from previous approaches of creating natural language descriptions of a conceptual model in Dalianis (1989), where the constructed natural

language generation system was criticized for generating a set of unordered sentences. A discourse
is also necessary to use to explain the overall semantics of the conceptual model, while each part of the conceptual model is related to one or many other parts of the model.

A number of appropriate discourses is proposed to answer the questions which are supposed to be posed by the users of a natural language generation system. The proposed discourses are analyzed with Hobbs' coherence relations (Hobbs, 1985; 1990) and a discourse grammar will be generalized from the discourses.

The discourse grammar and the methodologies of previous constructed NLG-systems, is used to propose a natural language discourse generation system for a conceptual model. The generation in the NLG-system is carried out at deep level and not a surface level, (Which is explained later). The discourse is created using a subset of Hobbs' coherence relations, (Hobbs, 1985; 1990).

Other problems which are also discussed are how to select the appropriate information from the conceptual model to create a suitable discourse. The selection is based on heuristics and the delimitation is carried out with user modeling. i.e to model the knowledge level of the user. A suggestion of a user modeling method is proposed. The method is inspired by the method in Finin (1989), where stereotype users where structured in a hierarchy. The application of the user models on the selected discourses is based on some ideas in Paris (1985; 1988).

2. Support systems for information and data modeling

2.1. The information system development process

The purpose of an information system is to store and retrieve information about a real world domain. During the construction of an information system various small problem can emerge. To avoid flaws in later stages of the information system development process, an information system has to be described at a higher level abstraction than at the programming level. For this purpose various high level and abstraction languages have been developed. One of them is the Conceptual MOdeling Language, (CMOL), (Bubenko & Lindencrona 1984; Bubenko 1986), in which a conceptual model can be expressed. A conceptual model, (CM), describes a piece of the real world, a domain, in an unambiguous and non-redundant way.

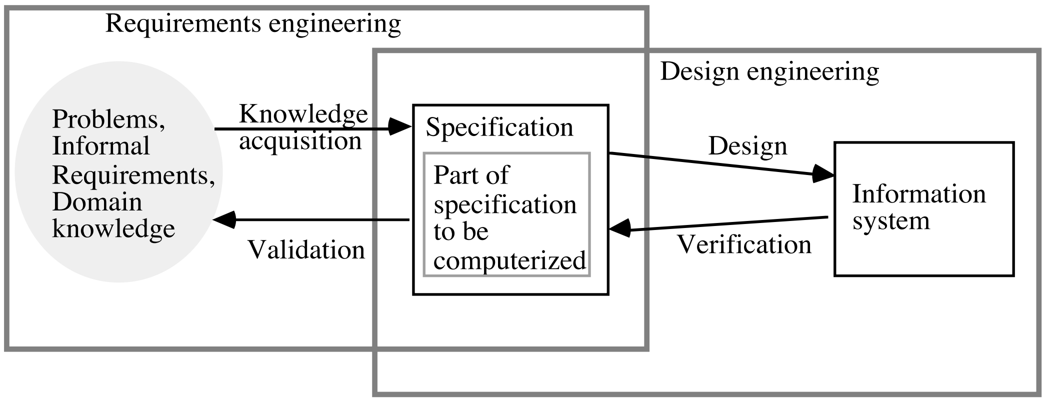

Fig. 1 Information Systems Development

A conceptual model is built and revised during a comparatively long period of time or different period of times. To build a conceptual model of a system or an organisation is by necessity an iterative process ranging from a vague idea of what it will do to a full fledged system. But even when the computer system is fully developed new extensions of the system will be needed in order to cope with changes in the domain (real world). Within the field of Information Systems, the development process can be said to be divided into *requirement*, and *design engineering*. Requirement engineering is discussed in Hagelstein (1988). Requirement engineering defines the method and effort to construct a specification in a domain, as a solution of a real world problem, i.e. a conceptual model of a computer system supports the domain.

Requirement engineering consists of two major parts: *Knowledge acquisition* and *Validation*. Knowledge acquisition consists of gathering information about the known problems, informal requirements and domain knowledge. From this information a specification is built. The validation consists of rephrasing the specification back to the domain expert to find out if it correctly describes the domain, to execute and test the model or to analyze it with heuristic methods.

Design engineering consists of two parts: *Design* and *Verification*. This paper does not cover Design engineering, since it does not affect the conceptual model.

2.2. A Conceptual model

According to Boman et al, (1991), Bubenko & Lindencrona (1984) and Bubenko (1986), an information system contains a *conceptual model* and *an information processor*. A conceptual model, (CM), contains a conceptual schema, (CS) and *an information base* or a fact base. The conceptual schema can be considered to be a *skeleton* description of the real world or domain, i.e. which entities possible can exist. The information base contains statements describing an object system, i.e. the domain or real world. The purpose of the information processor is to enable users to query and update the conceptual schema and the information base.

A conceptual model consists of a static and a dynamic part, of entity types, (objects) , relations, attributes, ISA-links, events, Static- and Dynamic integrity constraint rules and Derivation rules, (SDD-rules), and finally the information base contains instances, (facts).

2.2.1. Validation and support tools for validation

The validation process checks if the constructed conceptual model is correct according to the real world. The validation process shall detect flaws and give suggestions for correction. This can be achieved by analyzing the conceptual model with expert systems, by doing consistency checking (Kuntz & Melchert, 1989; Tauzovich, 1989; Wohed, 1988), by executing the conceptual model in e.g. MOLOC, (Johannesson, 1990), or by paraphrasing the conceptual schema back at single sentence level to the user for validation, in e.g. AMADEUS, (Black, 1987) or in Dalianis (1989). The graphic representation of the domain is also a part of the validation tool, ALECSI, (Cauvet et al, 1991) RIDL*, (De Troyer et al, 1988) Validation is one of the most important tasks in the requirements engineering, since the validation will reveal if something has gone wrong in the conceptual modeling phase. In this phase the domain expert will make his judgement if the conceptual model is the one he intended.

A method of validation is to paraphrase the model into natural language, (NL). NL is a reference for all people involved in the development of the system. The paraphrasing is usually performed by the system analyst, but it would be convenient if it could be carried out automatically, so the domain expert himself could validate the conceptual model without having deeper knowledge in the conceptual modeling formalism.

None of the support tools mentioned above has any natural language discourse generation component.

2.2.2. Prototyping

MOLOC stands for MOdeling in LOgiC, (Johannesson, 1990), which is a prototype semantic database management system. MOLOC is a support system for conceptual modeling, where you can design and execute a conceptual model of a database. MOLOC demonstrates rapid prototyping and testing without having to construct the real system. MOLOC has a graphical interface called MGI, (MOLOC-Graphical Interface). You can design your conceptual schema in MGI, but the SDD-rules and the events updating the CM have to be stated directly in the MOLOC formalism.

2.2.3. The different users

One can distinguish three groups of individuals who are using conceptual models and could consequently use a natural language generation system: the domain expert, the system analyst and the end user. They have different needs and knowledge.

1) The *domain expert* , (DE), will need the model paraphrased to check if everything is correctly represented, if all facts are present, if the concepts have correct names and if the model is logical.
2) The *system analyst,* (SA), is interested in the function of the conceptual model for building a computer system and if the purpose of the model is correct.
3) The *end user,* (EU), wants to get a quick overview of the model to know how the knowledge is stored and how to navigate in the system.

Here we have separated three users of the conceptual model and three various types of information which need to be paraphrased into natural language.

There are various reasons for paraphrasing a CM to NL:

- To lower the conceptual barrier of the user.
- To ease the understanding of the CM-formalism for a DE.
- To give possibility for a DE to validate the model by himself.
- To ease the understanding of the domain for a SA.
- To detect errors and traps in the CM.
- To focus on certain aspects of a CM.
- To have a reference language (NL) which the DE, SA and EU understands.
- To teach the conceptual model formalism for a DE or a SA.
- To introduce a newly assigned person to the domain.
- To give a quick overview in the beginning of the conceptual modeling phase where the persons involved in the modeling phase need to know what has been modeled until now.
- To inform an end user of a natural language interface to a database how the database is organized and which questions s/he can ask to obtain information from the database.
- To indirectly validate the CM by the dictionary writing for the NLG-system.

The generation will be divided and combined between generated information from both the CS, which describes the type of information, and how it is stored in the database, and corresponding instances from the database. This combination will help the system engineer to validate the CM and the end users to navigate in the computer system.

The natural language generation can do a sort, selection or enumeration of, for example, subclasses, which will help the DE to remember if he has forgotten anything.

The generation from a CM can be performed at two levels: one at a general conceptual schema level and the other on a conceptual model level, where also instances of objects are used.

The set of questions and commands the system should handle :

- What do you know ?
- Describe an entity type !
- Describe instances of entity types !
- What is the relationship between different entity types ?
- What is the relationship between different instances ?
- What events are there ?
- What SDD-rules constraint the CM ?
- Which entities types are affected by which event ?
- List all entity types, (events, SDD-rules, instances) !

2.3. Various proposed discourses

The examples on proposed discourses below, originates from both the author's work, (Dalianis, 1992) and the users of the conceptual modeling tool MOLOC and MGI. The conceptual model below fig. 2 is used for the content of the proposed discourses. The text examples should help the system analyst, (SA), the domain expert, (DE) and the end user, (EU) in their various tasks.

A conceptual model in the car domain

Car schema

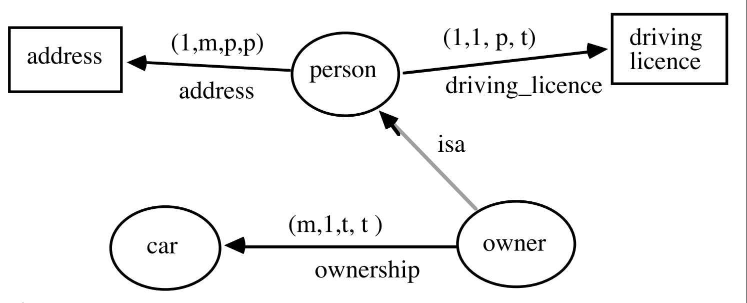

Fig 2.

Car information base

```
person(carl).
driving_licence(carl,121).

person(lisa).
address(lisa,211).

owner(robert).
address(robert,311).
driving_licence(robert,321).

car(volvo).
ownership(robert,volvo).
```

What do you know ? (A sort of help function)

What do you know ?

*list all ** (* stands for parts of the CM)
entity types
attributes
events
relationships between different entity types
relationships between different instances
relationships between entity types and events
SDD-rules
  Static integrity constraint rules
  Dynamic integrity constraint rules
  Derivation rules.
facts:
instances of entity types

List all [A] questions (The user□will get an overview of the contain of the CM)

List all entity types !

person
owner
car

What-is-[A] ? questions (The user asks about an entity type or instance)
<u>What is a person ?</u>
1) *Some persons are owners*
A person can have exactly one address
and exactly one driving licence
Carl is a person.
He has a driving licence 121 and <u>no address</u>
Lisa is a person
She has an address 211 <u>and no driving licence</u> (Informing about non-existent facts is optional)

<u>What is Carl ?</u>
2) *Carl is a person.* Facts
He has a driving licence 121 and <u>no address</u>
A person can have exactly one address Schema (This part is optional)
and exactly one driving licence

<u>What does following SDD-rule mean ?</u>	

The Static rule expressed in MOLOC
inconsistent :-
ownership(P,C),
not(address(P,A)).
	In Logic
	IF P owns C AND
	NOT P has_address A
	THEN inconsistent

The inconsistency means that the static integrity constraint rule is false
3) *Every owner (person) who owns a car must have an address.*

What-is-the-relation-between [A] and [B] and [C....] questions
<u>What is the relation between a owner and a car ?</u>
4) *Every owner is a person,who must own one or more cars and*
every owner must have an address
Schema S-rule

Events
Here follows questions which concern the dynamic part of the model.
<u>What events are there on cars ?</u>
buy_car Enumerating

<u>What are affected by the event buy_car ?</u>
buy_car affects:
car
person
owner
address
and
a static integrity constraint rule

<u>What are the relations between the event buy_car and car, person, owner, address ?</u>
5) *If a person buys a car*
then he must became an owner
Every person who owns a car must have an address.

An execution with MOLOC, (Johannesson, 1990), enhanced with a proposed NLG-system which gives explanation of what is carried out during an event is shown below.

The user executes an event.

<u>Which event to perform ?</u>

Let Carl buy a car VOLVO !
6) *If you let Carl buy the car VOLVO*
then the totality between attributes will be violated
A car must be owned by exactly one owner
Carl owns the car VOLVO
Robert owns the same car VOLVO

The user executes an event.

Which event to perform ?
Let Carl buy a car SAAB !
7) *If you let Carl buy a car*
then the static integrity constraint rule will not hold which says
For a person to became an owner of a car s/he must have an address
Carl has no address

All together there are 41 proposed discourses.

These proposed ideas are partly implemented on sentence level in AAIS-Prolog on a Macintosh, (Dalianis, 1989). The sentence level translation has also been discussed in Chen (1983).

The question types could, for example, be selected from menus, and the objects by pointing with a mouse on a graphical conceptual schema. The input to the question types could be extended with some limited text input.

3. Text generation

3.1. Discourse structure and analysis

The syntax and semantics of sentences have been well-studied and the syntax of a sentence is well defined. Previous constructed systems which generated natural language sentences from conceptual models are described in Dalianis (1989; 1990), however, given that the information contained in a conceptual model is context dependent, there has been a demand to describe the relationships between natural language sentences, i.e. a discourse which is a set of related and interconnected natural language sentences.

If we look at a discourse, we know that its sentences are more loosely kept together, than the parts of the sentences themselves. For example, if we mix a set of sentences in a discourse we would probably still understand the message but this would require a large effort and some information would of course be lost. A discourse which is easy to understand with less effort is called *coherent*.

There are many methods for analyzing discourses and understanding how sentences are connected. The main principle is to find so called key words or rhetorical primitives which are described in Rhetorical Structure Theory, (RST), (Mann, 1984; Mann & Thompson, 1988) or the coherence relations of Hobbs, (Hobbs, 1985; 1990). The coherence relations, for example, relate two or more sentences to one unit, and this unit in turn is ordered in a higher hierarchical structure, which describes the entire discourse.

The assumption made is the following: A discourse is coherent if its sentences can be fitted into one overreaching relation.

Hobbs' coherence relations are explained below:

Occasion relations	
occasion	a weak causal relation, a coherence between events in the world.
cause	special case of the occasion relation, the normal causal relation (keyword *if ..... then..*).
enablement	special case of the occasion relation, the first assertion enables the second assertion.
Evaluation relations	
evaluation	a meta comment, (keyword e.g. *Do you understand so far... This is good news*)
Ground-figure and explanation relations	
ground-figure	also called background, it is old information, background information, often time related and related to new information.
explanation	is an inverted cause, i.e. a proposition is caused by something. (keyword *because*)
Expansion relations	
elaboration	describes an object or event more in detail, (keyword *i.e. that is)*
exemplification	gives an exemplification of an type of event or object (key word *for example*).
generalization	a proposition is generalized, (keyword *it is well known that...*) it is the same as exemplification, but the order is switched.
parallel	two or more sequential propositions at the same level describing the same object or event level.
violated-expectation	two different assertions gives two different results a proposition is true but... (keyword *but*).
contrast	two similar assertions gives two completely different results.

Fig. 3 Hobbs' coherence relations

3.2. Text generation technology

In the research field of text generation or natural language generation there exist various approaches to solve the problem of generating appropriate natural language text. See fig. 4.

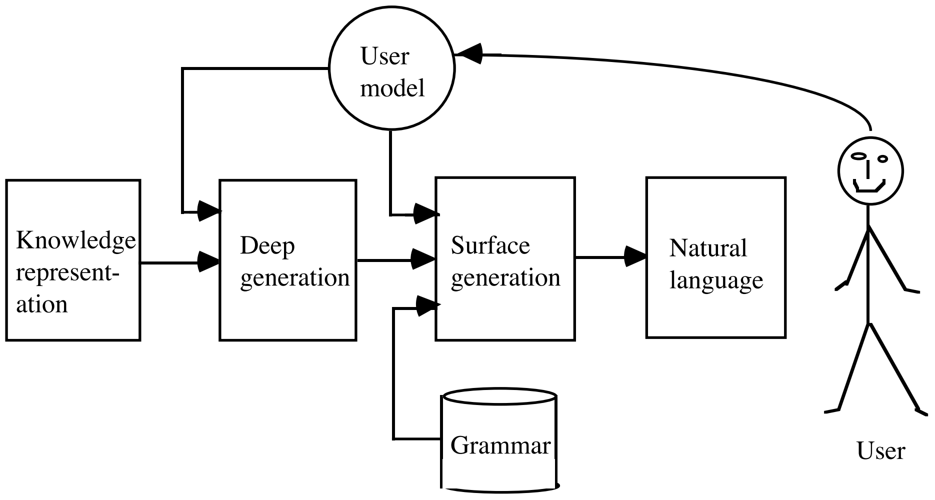

Fig. 4 An "average" text generation system

3.2.1. Deep and surface generation

Many researchers consider the task of natural language generation from a computer to consist of two sub-tasks, namely:

1) Deep generation
2) Surface generation

In the first subtask, the deep generation, it is decided what to say from the abundant knowledge base. The planning and organization of the information content is determined. Thus it is concluded in what form it should be presented according to a specific user model and in which order should the sentences be generated to make the text coherent. The deep generation in a computer must make steps similar to when a human generates text. During the second subtask, the surface generation, it has to be decided how to say it, i.e. the realization of the syntactic structures. Moreover a selection of the lexical items appropriate to express the content has to be made. This paper concerns the first subtask, the deep generation component of the natural language generation system.

3.2.2. Deep generation

No one has yet enumerated the kind of tasks a text planner should be able to do, but some of the problems are known. One problem is the content determination, i.e. to select what to say of the abundant information in a knowledge base. A partial solution to this problem is dependent on the question and the knowledge level of the user.

There are various approaches for solving these problems, ranging from discourse strategies as in the system TEXT , (McKeown, 1985a; 1985b), applications in RST, (Hovy, 1988; Rankin, 1989), theorem proving approaches in the KAMP system, (Appelt, 1985), to user modeling approaches (Paris, 1985; 1988; Kass & Finin, 1988; Finin, 1989).

3.3. Summary

The problem for many users of the conceptual models is to have an overview and understanding of the represented concepts, it therefore seems obvious to translate the model into natural language. Since the conceptual model is heavily dependent on all its parts, it seems appropriate to have a natural language discourse generation. This can be achieved by connecting each part of the conceptual model with the coherence relations of Hobbs, (Hobbs, 1985; 1990). This will make the natural language generation produced from the system coherent and easily read.

A technique for natural language generation is available and discussed in Dalianis, (1990) and the problem is to find what parts of the conceptual model corresponds to which coherence relations. The sentence level generation has already been carried out by the author and described in Dalianis (1989).

4. System overview

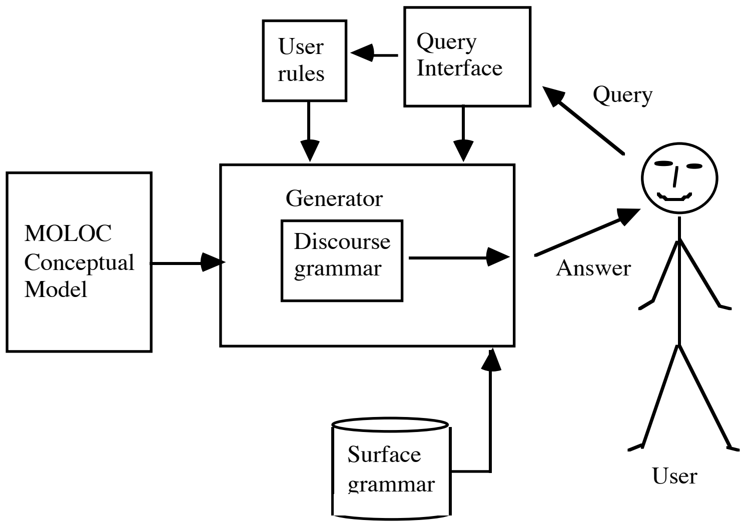

Fig 5. Overview of the proposed natural language generation system

The system is built around three modules: *the user rules, the discourse grammar* and *the surface grammar,* with a query interface which processes the question input from the user. The query interface passes the processed input both to the user module and to the generator. The user module will find the intentional goals of the user and at what level the user is by the way s/he is asking questions, this is described in (Dalianis, 1992) This information together with the query from the user will also give an answer to the user's first intention: to reply to the user's question by making a selection of information from the knowledge base.

The generation is carried out in three steps:

1) The user rules: These are used for finding out what the user knows and building a dynamic user model which helps to select the correct information from the conceptual model.
2) The discourse grammar: This builds a discourse structure from the selected information. The discourse structure should fulfil the intentions and goals of the user.
3) The surface grammar is not described here. It is at a syntactic level and belongs to the surface generation.

4.1. Extracting discourse grammar rules.

An discourse analysis with Hobbs' coherence relations is carried out below, on one of the 41 previous proposed texts in chapter 2, namely text nr 1).

<u>What is a person ?</u>
1) Some persons are owners
2) A person can have exactly one address and
3) (A person can have) exactly one driving licence
4) Carl is a person.
5) He has a driving licence 121
(6) and (he has) no address)
7) Lisa is a person
8) She has an address 211
(9) and (She has) no driving licence)

The text above gives the discourse tree, shown in fig 6. (The numbers in the discourse tree correspond to each sentence above, sentences 6 and 9 are not included in the analysis)

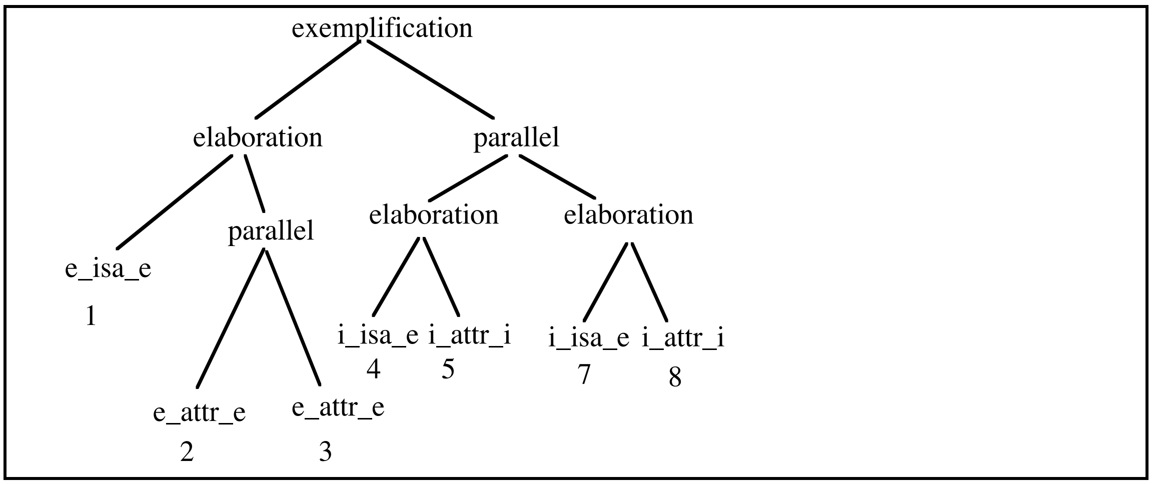

Fig. 6 The main division of the text is the exemplification relation between the schema and instance level. The schema leaf is divided by the elaboration relation which elaborates the ISA-relation into two equivalent attribute statements related by a parallel relation. In the instance leaf we find a parallel relation which describes two pieces of a discourse at the same level.

Each discourse is described by an elaboration relation which elaborates an ISA-statement into an attribute statement at ISA-level.

All the 41 proposed discourses were analyzed with Hobbs' coherence relations and it was found that only the *exemplification, generalization, elaboration, cause* and *explanation* relations were necessary for the analysis.

Discourse grammar

The discourse structure will focus on certain aspects of the conceptual model, (CM), and will linearize the parallel message of the conceptual model, this is also described in (Dalianis, 1992). The following discourse grammar is generalized from the analysis performed on the proposed texts. The discourse grammar uses the coherence relations of Hobbs, the ones which were used in the analysis of the proposed texts. The terminals of the discourse grammar are single sentences or elements of the CM . The discourse grammar will transform a piece of a conceptual model to a discourse structure which then easily can be transformed to a text by a surface generator.

The representation below is in Backus-Naur form or a Context Free Grammar. "{}" means that something is optional and " | " means or and "( )*" means none, one or many times.

The discourse grammar definition

Top level

```
DISCOURSE           ::==    EXEMPLIFICATION
DISCOURSE           ::==    {GENERALIZATION}        (optional)
DISCOURSE           ::==    ELABORATION {GENERALIZATION}
DISCOURSE           ::==    CAUSE
DISCOURSE           ::==    EXPLANATION
```

Rest

```
ELABORATION         ::==    E_ISA_E PARALLEL
ELABORATION         ::==    I_ISA_E PARALLEL
ELABORATION         ::==    I_ISA_E I_ATTR_I
ELABORATION         ::==    E_ISA_E E_ATTR_E
ELABORATION         ::==    E_ATTR_E ELABORATION
ELABORATION         ::==    EVENT_NAME  CAUSE
ELABORATION         ::==    PARALLEL EXPLANATION
ELABORATION         ::==    PARALLEL PARALLEL

EXEMPLIFICATION     ::==    E_ISA_E PARALLEL
EXEMPLIFICATION     ::==    E_ATTR_E I_ATTR_E
EXEMPLIFICATION     ::==    ELABORATION ELABORATION
EXEMPLIFICATION     ::==    ELABORATION PARALLEL
EXEMPLIFICATION     ::==    CAUSE PARALLEL

EXPLANATION         ::==    CAUSE ELABORATION
EXPLANATION         ::==    PARALLEL E_ISA_E
EXPLANATION         ::==    CAUSE I_ATTR_I
EXPLANATION         ::==    SUCCEED_FAIL S_RULE

GENERALIZATION      ::==    ELABORATION PARALLEL
GENERALIZATION      ::==    ELABORATION ELABORATION

PARALLEL            ::==    ELABORATION ELABORATION
PARALLEL            ::==    I_ISA_E I_ISA_E
PARALLEL            ::==    E_ATTR_E E_ATTR_E (E_ATTR_E)*
PARALLEL            ::==    E_ATTR_E E_ATTR_E (CAUSE)*
PARALLEL            ::==    I_ATTR_I I_ATTR_I (I_ATTR_I)*
PARALLEL            ::==    S_RULE S_RULE
PARALLEL            ::==    PRECONDITION PRECONDITION
```

CAUSE	::==	E_ISA_E S_RULE
CAUSE	::==	E_ATTR_E S_RULE
CAUSE	::==	E_ATTR_E E_ISA_E
CAUSE	::==	EVENT EXPLANATION
CAUSE	::==	PARALLEL S_RULE
CAUSE	::==	ELABORATION S_RULE

Terminals are single sentences or parts of the CM

E_ISA_E	::==	ENTITY TYPE1 ISA ENTITY TYPE2 (schema)
I_ISA_E	::==	INSTANCE ISA ENTITY TYPE (mixed)
E_ATTR_E	::==	ENTITY TYPE1 ATTR ENTITY TYPE2
I_ATTR_E	::==	INSTANCE ATTR ENTITY TYPE
I_ATTR_I	::==	INSTANCE ATTR ENTITY TYPE
EVENT	::==	event name or type of event
PRECONDITION	::==	precondition in an event
(SUCCEED_FAIL)	::==	(the rule will succeed)|(the rule will not hold)
(SUCCEED_FAIL)	::==	(the attributes will hold)|(the attributes will not hold)
S_RULE	::==	(STATIC | DYNAMIC) rule
D-RULE	::==	DEDUCTION rule

The above discourse grammar describes the connection between a conceptual model and a discourse form. This means that the user's question together with the available conceptual model will create a discourse according to Hobbs' classification of discourse structure.

The discourse grammar is almost executable as it stands in Prolog, only some minor syntactic changes are needed. The problem is then that the grammar will overgenerate and that it does not have a control mechanism. This control mechanism will be created by using the same method which was used for analysing the discourses and by implementing features and control predicates.

4.2. The discourse grammar and the control of the generation

The implementation language is Prolog, which is well-suited both for parsing and for generation of natural language. A first implementation using a subset of the discourse grammar has been carried out. The discourse grammar is a so called Definite Clause Grammar, DCG, (Pereira & Warren, 1980; Clocksin & Mellish, 1984) and is executed and controlled by the Prolog interpreter. The execution of the grammar is performed backwards. The terminals consist of the selected information for each question. Each terminal consists of a single sentence at either schema, instance or mixed level. Further more various *features* and *predicates* of the grammar are implemented to control the generation.

The *features* with the values i,e,r,_ for describing entity types and instances of them.

i stands for instance level
e for entity type or schema level
r for rule
_ for the anonymous variable or irrelevant.

The coherence relation *exemplification* extended with features is an example of the above control of the grammar. The relation exemplification is a divider between explanation at the schema level, and explanation at the instance level i.e. features entity type e or instance i. Another example is the *elaboration* relation which can be performed both at schema and instance level, but it has to be kept either of the ways. These two cases are seen in the extract of the discourse grammar below.

Predicates

The predicate *not_occur/2* checks whether any part of the terminals are used more than once for generation, i.e. none of the terminals occur more than once in the discourse tree.

The predicate *same/2* is used to check if two clauses has any connection to each other at all, for example in elaboration, to talk about the same entities.

When a piece of the discourse tree is generated it is checked for any violations of the generation rules.

Extract from the discourse grammar which generates the discourse below

```
discourse(exemplification(E))
        --> exemplification(_,E).
exemplification(_,(elaboration(E) & parallel(P)))
        --> elaboration(e,E), parallel(i,P),{not_occur(E,P),!}
elaboration(i,i_isa_e(I1) & i_attr_i(I2))
        --> [I1,I2],{i_isa_e(I1), i_attr_i(I2),not_occur(I1,I2),same(I1,I2)}.
elaboration(i,i_isa_e(I) & parallel(P))
        --> [I],{i_isa_e(I) , parallel(i,P),not_occur(I,P)} .
parallel(e,(e_attr_e(E1) & e_attr_e(E2)))
        --> [E1,E2],{e_attr_e(E1), e_attr_e(E2),not_occur(E1,E2),same(E1,E2)}.
parallel(i,elaboration(I1) & elaboration(I2))
        --> elaboration(i,I1), elaboration(i,I2),{not_occur(I1,I2),!}.
```

Example of a generation

Question: *What is a person ?*

```
?- list_db.                                   selected sentences
[some,persons,are,owners]
[carl,is,a,person]
[lisa,is,a,person]
[carl,has,an,driving_licence,121]
[lisa,has,an,address,211]
[a,person,can,have,exactly,one,address]
[a,person,can,have,exactly,one,driving_licence]
  yes
?- discourse(TREE,NL).
  TREE = exemplification(elaboration(e_isa_e(
                              [some,persons,are,owners])
                             &
                             parallel(
                                e_attr_e(
                                   [a,person,can,have,
                                   exactly,one,address])
                                  &
                                  e_attr_e(
                                    [a,person,can,have,
                                     exactly,one,
                                     driving_licence])))
                    &
                    parallel(elaboration(
                             i_isa_e([carl, (is),a,
                                     person]) &
                              i_attr_i(
                                 [carl,has,a,
                                  driving_licence,121]))
                           &
                           elaboration(
                             i_isa_e([lisa, (is),a,
                                      person]) &
                               i_attr_i(
                                 [lisa,has,an,address,
                                  211])))),
discourse 1)
        NL =  [[ some,persons,are,owners],
               [a,person,can,have,exactly,one,address],
               [a,person,can,have,exactly,one,driving_licence],
               [carl, (is),a,person],[carl,has,a,driving_licence,121],
               [lisa, (is),a,person],[lisa,has,an,address,211]]
```

The example above gives an idea how it would technically be possible to achieve the above proposal and how the discourse tree would look like.

User modeling

One idea for delimiting the amount of generated text is to have a user model, such that the question of the user will trigger two actions: First to update the user model and second to select information from the knowledge base, i.e. the conceptual model, to answer the question. This is discussed in detail in (Dalianis, 1992)

The selected user model will be instances of a user theory. Thus a set of rules will be triggered by a set of inputs from the user and consequently a user model will be built. The user modeling system described below will give us a dynamic user model such that the answers will be dependent on the current dialogue with the user and level of the user.

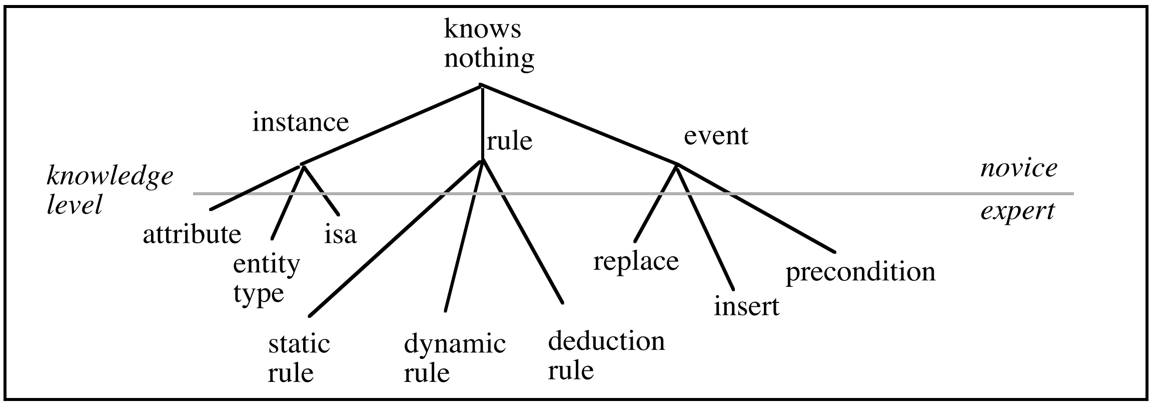

Fig. 9 Concept generalization hierarchy of the conceptual modeling language structure

A modified version of the hierarchy which models various users in Finin (1989) will be used for giving a description of the surroundings of the conceptual schema to the user in the right knowledge level. In the hierarchy we will use a rough division between novice and expert users, but the hierarchy could also be made finer grained with additional user levels. For example: *beginner* and *intermediate user*. It may be common that a user may have local expertise in one area but be a novice in another area.

What we model is the user's knowledge of the existence of various concept in the conceptual modeling language. We can consider that the knowledge of concepts above the knowledge level is the novice's knowledge and the knowledge of concepts below the knowledge level is the expert's knowledge. But of course knowledge in expert concepts implies knowledge in novice concepts.

The knowledge level is a level which indicates a point, at which a novice user becomes an expert user. The knowledge level is drawn in this way because of the division between the general concepts known by a domain expert or an end user and the conceptual modeling formalism known by an expert. The knowledge level is not experimentally tested and is rather arbitrary. It could be adjusted empirically.

5. Conclusions

Support tools for conceptual modeling lack natural language generation functions. In this paper we have argued for the need of natural language generation as a support tool for conceptual modeling.

The paper proposes a set of appropriate questions which could be posed by the user and a set of suitable natural language discourses to answer these questions. From the proposed discourses a discourse grammar is generalized. The discourse grammar connects a conceptual model to a discourse structure. A natural language generation system built on this grammar is suggested. The goal of the system is to improve the validation of a conceptual model.

The purpose of a natural language discourse in this setting is to answer a question from a user in a more satisfying and contextually sensitive way than a single sentence or a set of unordered sentences would. We know that natural language lowers the conceptual barrier for the user and that a natural language discourse gives a better comprehension, since the receiver of the discourse when reading the linear text will try to identify the higher order structure of the text, according to Anderson (1985), and consequently the conceptual model.

Future research will be to implement the proposed system and to extend the grammar for more cases and then test the system to determine which discourse structures the users require.

I would conclude with: Interpreted data gives information, reasoning about information gives knowledge and knowledge expressed in natural language gives understanding !

Acknowledgements

I would like to thank my advisor Carl Gustaf Jansson and my thesis committee: Janis Bubenko, Carl Brown and Östen Dahl for generously contributing of their knowledge in their fields and for their valuable comments. I would also like to thank Paul Johannesson and Rolf Wohed and others in the SYSLAB research group for interesting discussions which contributed to this paper and thank, Guy Ridley, for checking the English.

6. References

Anderson J.R. 1985. Cognitive Psychology and Its Implications, Carnegie-Mellon University, W.H. Freeman and Company .

Appelt, D.E. 1985. Planning English Sentences, Cambridge University Press.

Black, W.J. 1987. Acquisition of Conceptual Data Models from Natural Language Descriptions, In The Proceedings of The Third Conference of the European Chapter of Computational Linguistics , Copenhagen, Denmark.

Boman, M. et al, 1991. Conceptual Modeling, Department of Computer and Systems Sciences, Stockholm University.

Bubenko, J. & Lindencrona, E. 1984, Konceptuell modellering - Informationsanalys, Studentlitteratur, Lund, (in Swedish)

Bubenko, J. 1986. Information System Methodologies - A Research View, SYSLAB Report no 40, Department of Computer and Systems Sciences, Stockholm University, Sweden.

Cauvet, C. et al. 1991. ALECSI: An expert system for requirements engineering, in Proceedings of Computer Aided Information System Engineering, CAISE-91, Eds. R. Andersen et al, Trondheim.

Chen, P. P-S. 1983. English Sentence Structure and Entity Relationship Diagrams, Information Sciences 29, p.p. 127-149.

Clocksin, W.F. & Mellish, C.S. 1984. Programming in Prolog, Springer Verlag.

Dalianis, H. 1989. Generating a Natural Language Description and Deduction from a Conceptual Schema, SYSLAB Working Paper no. 160, Department of Computer and Systems Sciences, Royal Institute of Technology.

Dalianis, H. 1990. Deep generation strategies and their application for creating alternative descriptions form conceptual schemas, SYSLAB Working paper no. 177, Department of Computer and Systems Sciences, Royal Institute of Technology.

Dalianis, H. 1992. User adapted natural language discourse generation for validation of conceptual models, (licentiate thesis), SYSLAB Report no 5, Department of Computer and Systems Sciences, Royal Institute of Technology.

De Troyer, O. et al, 1988. RIDL* on the CRIS case: A workbench for NIAM,Computerized Assistance During the Information Systems Life Cycle.T.W Olle, et al. (eds).Elsevier Science Publishers B.V North Holland.

Finin, W.F. 1989. Gums - A General User Modeling Shell, in User Modeling in Dialog Systems (eds) A Kobsa , W. Wahlster, Springer Verlag.

Hagelstein, J. 1988. Declarative approach to information systems requirements, Knowledge Based Systems, Butterworth, Vol 1, No 4, 1988.

Hobbs, J.R. 1985. On the Coherence and Structure of Discourse, Report No. CSLI-85-37, October 1985.

Hobbs, J .R. 1990. Literature and Cognition, CSLI Lecture Notes Number 21, Center for the Study of Language and Information.

Hovy, E.H. 1988. Planning Coherent Multisentential Text, Proceedings of the 26th Meeting of the ACL, Buffalo, New York, 1988.

Johannesson, P. 1990. MOLOC: Using Prolog for Conceptual Modeling, Proceedings of the International Conference on Entity-Relationship Approach, North Holland.

Kass, R. & Finin, T. 1988. Modeling the User in Natural Language Systems, J. of Computational Linguistics, Vol 14, No 3, Sept 1988.

Kuntz, M. & Melchert, R. 1989. Ergonomic Schema Design and Browsing with More Semantics in the Pasta-3 Interface for E-E DBMSs, Proceedings of the 8th International Conference on Entity-Relationship Approach, Ed F.H. Lochovsky, Toronto, Canada.

Mann, W. C. 1984. Discourse Structures for Text Generation, Proceedings of the 22nd annual meeting of the Association of Computational Linguistic, Stanford, CA,

Mann, W. C. & Thompson, S.A, 1988. Rhetorical Structure Theory: Towards a Functional Theory of Text Organization, In TEXT Vol 8:3, 1988.

McKeown, K.R. 1985a. Textgeneration: Using discourse Strategies and focus constraints to generate natural language text, Cambridge University Press.

McKeown, K.R. 1985b. Discourse Strategies for Generating Natural Language Text, Artificial Intelligence, vol 27 no 1, Sept 1985.

Paris, C. 1985. Description Strategies for naive and expert users, Proc. of the 23rd Annual Meeting of the Association of Computational Linguistics.

Paris, C. 1988. Tailoring Object's descriptions to a User´s Level of Expertise, J. of Computational Linguistics, Vol 14, No 3, Sept 1988.

Pereira, F.C.N & Warren,D.H.D, 1980. Definite Clause Grammars for Language Analysis - A Survey of the Formalism and a Comparison with Augmented Transition Networks. J. of Artificial Intelligence 13, 1980, pp 231-278.

Rankin, I. 1989. The Deep Generation of Text in Expert Critiquing Systems, Licentiate Thesis No 184, Linköping University, 1989.

Tauzovich, B. 1989. An Expert System for Conceptual Data Modeling, Proceeding of the Entity Relationship Approach Toronto, Canada.

Wohed, R 1988. Diagnosis of Conceptual Schemas, SYSLAB report no 56, Department of Computer and Systems Sciences, Royal Institute of Technology and Stockholm University 1988.