C$^{\text{O}}_{\smile}$L$^{\text{O}}$R-$\mathcal{X}$: Using Knowledge from WordNet for Conceptual Modeling

J.F.M. Burg* and R.P. van de Riet
Department of Computer Science
Vrije Universiteit
The Netherlands
jfmburg,vdriet@cs.vu.nl

1 Introduction

In this chapter we will describe the role WordNet [MBF+93] plays in our linguistically based conceptual modeling environment. The modeling method we are developing is called C$^{\text{O}}_{\smile}$L$^{\text{O}}$R-$\mathcal{X}$ which is an acronym for the **CO**nceptual **L**inguistically based **O**bject oriented **R**epresentation Language for **I**nformation and **C**ommunication **S**ystems (ICS abbreviated to **X**), [BvdR94], [BvdR95b], [BvdR95a] and [BvdR95c], and it consists of several, graphical, modeling techniques that are all created using knowledge from WordNet interactively. In our view, WordNet is a source of *reusable* knowledge, that can be used during conceptual modeling to ensure that the resulting models are correct. On the other hand, we are achieving a certain degree of consistency among the models and between the models and the original problem specification, through the paraphrasing of the models into Natural Language sentences. This natural language generation process is also supported by WordNet.

The COLOR-X models are used to bridge the gap between the requi-

*Supported by the Foundation for Computer Science in the Netherlands (SION) with financial support from the Dutch Organization for Scientific Research (NWO), project 612-123-309

rements analysis and software design phases of a software development process. By capturing the requirements in a adequate way, we are able to generate the more technical design models automatically.

This project is part of the LICS-project (Linguistically based Information and Communication Systems), in which we investigate how linguistic knowledge could be used when building Information and Communication Systems. In our view, the problem of controlling the *meaning* of words is becoming a key issue in developing Information and Communication Systems, like Database Management Systems, Communication Systems, Office Automation, etc. The systems offer highly sophisticated tools for efficient storage, processing and transmission of data. However, transmission of data is only useful when the sender and the receiver agree on the meaning of the words. Database retrieval is only successful when the user knows where to look and what words (s)he should use. Although Data Dictionaries have existed for decades, and the best among them do contain meaning definitions of terms [BvdRC93], they are of limited help, for several reasons worth noting. First, they lack any linguistic knowledge, which makes it difficult to sort out semantic and morpho-syntactic aspects. Secondly, they lack a formal basis, hence making it impossible for the system itself to reason with the meaning descriptions. Thirdly, they are usually *closed systems*, and not integrated with the user interface, the CASE design environment, or other applications.

LICS itself is a subproject of the LIKE-project (Linguistic Instruments in Knowledge Engineering) which is a consortium of researchers of three disciplines: Linguistics, Business Administrators and Computer Science. The LIKE-project is focusing research around the theme: how linguistic instruments can be used profitably in the area of Knowledge Engineering, see [vdR94]. Preliminary research projects have shown the

usefulness of this approach: a linguistically based data dictionary environment [BvdRC93], a linguistic interpretation of ER-models [BvdR92b] and a general feasibility study to add linguistic knowledge to the process of conceptual modeling [BvdR92a].

Before we will explain in which way we use WordNet in our COLOR-X environment, we will first give our reasons to incorporate linguistic knowledge into conceptual modeling. One of the main reasons to use linguistic knowledge is to make the use of the words appearing in the models consistent. Among the obvious rules are:
class names should be nouns and *relationship names should be verbs.*

Less trivial, but more interesting and important, are the rules between the meaning of words used in the model:
certain relationship-types require certain class-types and
certain class-types cannot be related in some systems.

An example of the first rule constrains the type of class required in the *buy*-relationship to be (a descendant in the is-a hierarchy of) *person.* An example of the other rule forbids a *marry*-relationship between two persons of the same sex.

Another reason to use linguistic knowledge in modeling techniques is to give more expressive power to those techniques. It will be shown further on in this paper that adding the *roles* objects play in a relation, such as *agent* or *instrument*, will make the model easier to understand and to use. The addition of modalities, like *must* and *permit*, clarifies the status of the used relationships.

An additional nice feature of a linguistically based modeling technique is that it is relatively easy to generate natural language sentences from it, in order to give feedback to the system designers and to the end-users as well. This feedback consists of generated sentences during

the modeling phase, in order to check if the model is consistent with the requirements. On the other hand this feedback consists of explanation facilities, similar to the way Gulla defines and uses in his PPP-case tool [Gul93].

The integration achieved by using linguistic knowledge in the software engineering process consists of two parts: *vertical* and *horizontal* integration. To achieve a certain degree of vertical integration, integration between different *phases* of the software engineering process (like analysis, design and implementation), we have mainly focussed on the correspondence of design models and analysis information. The reuse of knowledge from WordNet and the validation and verification of the design models, made possible by the linguistic base of the models, reduce the gap between these two phases considerably. The object-oriented paradigm, on which our approach is also based, facilitates this vertical integration in a natural way, by using the same concepts in each phase. Founding the different views modeled in the design phase on the same underlying formalisms, as will be shown in the next section, integrates the design models horizontally adequately, i.e. integration between several models in one phase of the software engineering process.

Before we describe the use of WordNet as a source for this linguistic knowledge, we shall first introduce the COLOR-X models.

1.1 C$^{O}_{\smile}$L^{O}R-$\mathcal{X}$

At the moment, COLOR-X consists of two kinds of models [BvdR95c], which are both founded by formal (logical) specifications, that have a linguistic base (this underlying specification language (CPL: Conceptual Prototyping Language) is defined in [Dig89] and [DvdR91]):

1. COLOR-X **Static Object Model (CSOM)** [BvdR94], and [BvdR95b], which shows objects and classes of objects, the relationships between them, and static constraints upon them as they occur in a certain problem domain (or: Universe of Discourse). This model is the central model because it defines the overall structure of the system to be built.

2. COLOR-X **Event Model (CEM)** [BvdR95a], which shows the dynamic aspects of an Information and Communication System, in which all the occurring events and their dynamic and deontic constraints and restrictions are listed and ordered in time.

Both modeling techniques will be introduced shortly in the next sections.

1.1.1 C☺LOR-$\mathcal{X}$ Static Object Model

The aim of the COLOR-X Static Object Model (CSOM), or object models in general, is to structure the static aspects of a system independently of the implementation environment. This means that we model the *logical* structure of the system, in which we define the stable, robust and maintainable (i.e. also extendible) elements of the universe of discourse, about which information should be kept in the implemented system. We will not give a complete introduction of the CSOM modeling technique, but we will explain it by giving an example, and describing the elements occurring in it.

The example of a CSOM model, built for a simplified book-circulation system of a library, can be found in Figure 1.1. Among the information contained in this figure, are the following relationships:

- zero or more users borrow at most seven books from a library
- zero or more users return at most seven books to a library

- novels and encyclopedias, that consist of more than zero parts, are subtypes of book
- a library sends reminders to zero or more users
- zero or more users pay fines to a library
- a library possesses zero or more books (including novels and encyclopedias)
- users possess one and only one pass
- users have a name and address

The CSOM consists of [BvdR94], [BvdR95b]:

- **objects**: an object is an entity, occurring in the Universe of Discourse (problem domain), like *book* and *user*, that is able to save a state (information) and which offers a number of operations (behaviour) to either examine or affect the state [Jac92]. They are drawn as boxes with rounded corners (objects do not occur in Figure 1.1 though)
- **classes**: represent templates for several objects and describe how these objects are structured internally. A class describes what is common for all the objects that are instances of this class. They are drawn as boxes in Figure 1.1
- **relationships**: denote some static relations between two or more classes or objects. We distinguish two subsets:
  - **intra State of Affair (intra-SOA or user-defined)**: an example is *'borrow'*. They are drawn as a line from class to class, containing

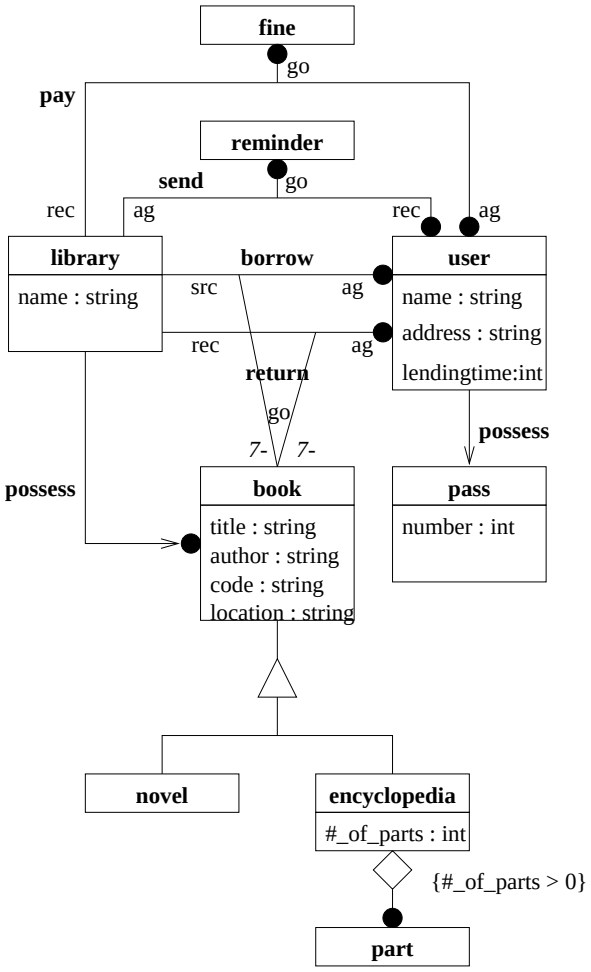

Figure 1.1: COLOR-X Static Object Model

1. **label**: has to be a verb (i.e. *borrow*)
2. **negation**: optional
3. **cardinality**: black dot (≥ 0), no dot nor a number (exactly one), or a number, constraining the number of related objects
4. **role**: agent, source, goal, etc. describes for each connected class the role it is playing in the relationship as a whole

– **standard**: inheritance (triangle on line), aggregation (diamond on line), possession (arrowed line), instantiation (dotted, arrowed line from object to class)

- **constraints**: on cardinality of relationships, on values of attributes, etc., occurring between braces.

1.1.2 C☺LOR-$\mathcal{X}$ Event Model

In general, the purpose of Dynamic Modeling, of which the COLOR-X Event Model (CEM) is an example, is to show the time-dependent behaviour of the system as a whole or a particular part of the system. A CEM is merely a trace of the events that could and should be performed in the Universe of Discourse. This way of modeling the dynamic aspects of the Universe of Discourse links up very well with the way these aspects are described in the requirements document. There exists however no automatic acquisition of conceptual models from these natural language sentences yet, as [Bla87] and [RP92] propose. The short introduction in CEMs will be analogous to the CSOMs one: we give an example, which is shown in Figure 1.2, and explain its elements.

The CEM consists of [BvdR95a]:

- **event-boxes**: drawn as boxes, containing:

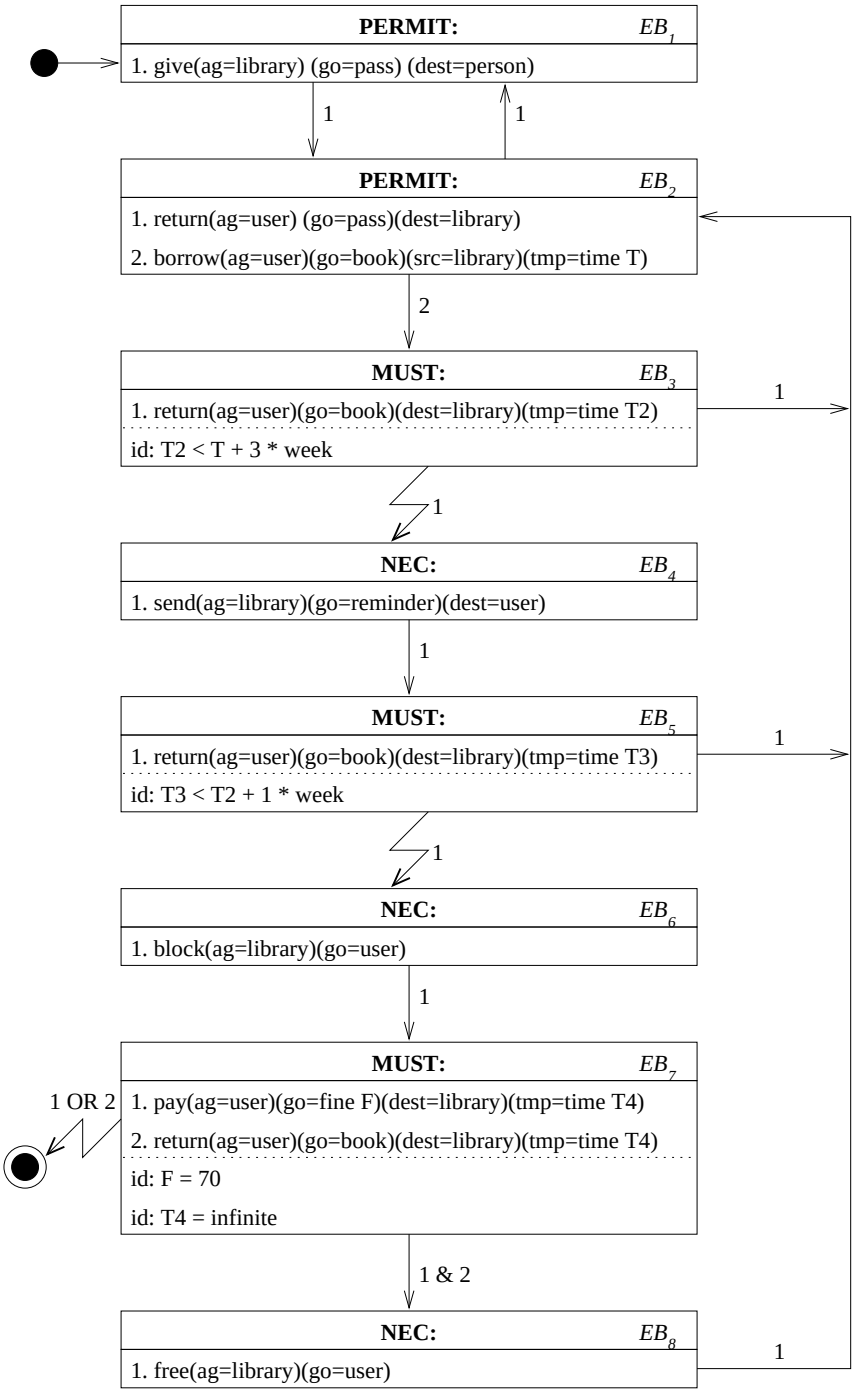

Figure 1.2: COLOR-X Event Model

- **modality**: permit, necessary, must
- **events**: numbered list of verbs and their objects and subjects, specified as CPL terms: [∼] Verb [([< $Card$[, $Distr$] >] **Role** = [Var **in**] Noun)]+
- **constraints**: predicates to which events should obey. These are shown below a dotted line

The event-box EB_1 of Figure 1.2 expresses the *permission of a library to give a pass to a person*

- **start and final node**: black dot and black dot with circle, respectively
- **actual occurrence of event**: drawn as an arrow between event-boxes, labelled with the numbers of the events occurring
- **no occurrence of event at all**: drawn as a lightning-arrow between event-boxes, labelled with the numbers of the events not occurring

A violation of a **NEC** event leads the system into an inconsistent state, and should therefore be prohibited by the system itself. **MUST** events may be violated, but the designer should specify the behaviour of the (external) entity or (sub-) system in the case of a violation. Because of the fact that there are three modalities which can be used (permit, necessary and must), there are three different kinds of event-boxes, the occurrences of which are triggered by words in the requirements document: for example *'can'* and *'is allowed to'* trigger a **PERMIT**-box and a **MUST**-box could be caused by the words *'has to'*.

This has led to the three standard building blocks, one for each modality. The events occurring in a **NEC** event-box cannot be vio-

lated so there are no outgoing lightning-arrows allowed. A **MUST**-box should always contain an *expiration-time* as a constraint, which may be infinite, in order to verify whether the obligation has been violated or not. A **MUST**-box should also have both a normal arrow and (violation) lightning-arrow for each event occurring in the box. A **PERMIT**-box may have a violation specification but this is not obligatory. By offering standard building blocks the resulting model could not be violating the graphical syntax rules. By checking if there exists exactly one start and one final state, and that every arrow goes from one block into another, we can verify if the model is syntactically correct.

One of the major novelties of the COLOR-X models, i.e. of both the CSOM and the CEM, is the possibility to handle and verify the *semantic* correctness of the models, by using WordNet on the one hand and by having a formal underlying representation on the other hand. By using WordNet as a source for reusable information and parts of the models we build our models out of semantically correct pieces, which may be adjusted to the needs of the modeler, but only according to certain rules. By adjusting and augmenting the models with additional or more specific information the semantic correctness can be guaranteed by formally describing the underlying relations between concepts occurring in models and the words that are allowed to be attached to these concepts [vdVvdR94]. To verify if the model as a whole is semantically correct and corresponds to the requirements as well, natural language sentences have to be generated and checked against the original requirements [BvdR94] or explanation facilities have to be included [Gul93].

The COLOR-X models are fully represented by formal CPL specifications, which are founded in several logics (like dynamic, deontic and temporal). Some other graphical conceptual modeling techniques

have this formal foundation as well. Because we formalize the *roles* objects play in a relationship, the *constraints*, the *modalities* of static and dynamic relationships and events as well as the relationships between *actions* and *events* [MT87], we are able to verify consistency and correctness inside and among the models. Additionally, we are able to make logical derivations out of these formalized specifications. The fact that both the static as well as the dynamic model have the same underlying formal representation integrates the models very strongly.

In contrast with, for example, OMT [RBP+91] we did not include a separate **Functional Model (DFD)** because we have developed an algorithm and implemented a prototype which retrieves this kind of knowledge from the CSOM and CEM models. We are still improving and refining both the algorithm and its implementation.

An additional difference with other modeling techniques, for example [FW93], is the absence of separate **object life-cycles**. The information captured in this kind of models is also generated out of the COLOR-X models automatically. Possible outputs that have been implemented include State-Transition Diagrams [BvdR95a] and Jackson Structure Diagrams.

2 WordNet supports Conceptual Modeling

The *process* of creating models according to the COLOR-X modeling method will not be treated entirely, but we will give a short overview of it. The main point of interest in this section is the support of, and the additional information retrieved from, WordNet.

To build a model of a certain universe of discourse we have to analyze it, retrieve the interesting information from it and possibly add some other knowledge. We assume the presence of a *requirements document*

that describes the universe of discourse and its problems (as well as proposed solutions). The next step is the analysis of this document, and the elicitation of the information to be put in the COLOR-X models. This results in lists of:

- **objects**, the entities playing an important role in the universe of discourse, that are identified by *nouns*
- **static relationships**, certain connections between two or more objects. We distinguish two kinds of relationships in this paper:
    1. **user-defined**, which are relationships introduced by the author of the requirements document. A typical example of this first kind would be *lend* in a library book-circulation environment
    2. **standard** or **conceptual** relationships that relate objects in a conceptual way, e.g. a book *is a* (kind of) document
- **events**, all actions and events that can, may, will or must happen in the universe of discourse

The elements of the first two lists form the basis of the COLOR-X Static Object Model (CSOM), the elements of the latter one, augmented with constraints and ordered in time, form the COLOR-X Event Model (CEM). We will treat each list separately, to show exactly what kind of information is (or can be) used from WordNet during the creation of conceptual models. We have implemented an environment that supports the selection of information from WordNet and its incorporation into the models. Examples of the implementations interface will be shown, when we are explaining the selection of information from WordNet in the following sections.

2.1 Objects

The elements of the **object** list, retrieved from the requirements document of an universe of discourse, are classified into **classes**, which contain objects with common properties and behaviour and include for the book-circulation department of a library: *book, library, user, reminder, fine, etc.* (see for example Figure 1.1). The classes identified here will occur in the CSOM model.

Because of the ambiguous nature of Natural Language, words could have several *meanings* (homonyms and polysemes), and many concepts are denoted by two or more words (synonyms). With the help of WordNet we try to find the right *word meaning* in each specific universe of discourse, see Figure 2.3. This approach is similar to the process of tagging pieces of text as described in the Chapter 'Building Semantic Concordances'. Polysemous word clashes should be avoided by replacing these words with other ones that have similar meaning. If this search is successful we know for the rest of the development process that we are using the right meaning and that communication about the models is not troubled by ambiguous word interpretations.

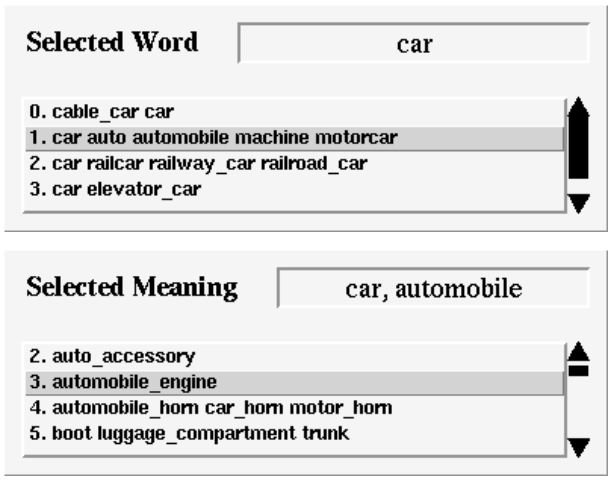

Figure 2.3: Specifying the meaning and attributes of 'car'

If we know which *concept* is meant by each word, we are able to retrieve more general information about it from WordNet, using WordNets *substance*, *part* and *member meronymy* relationships. Examples of these relationships are *concrete has substance cement*, *car has a engine* and *person is member of people*. The meronymy information will end up as aggregation, possession or attribute relationships in the CSOM, if they are useful and meaningful in the universe of discourse that is considered, The part and member meronymy information will be translated into aggregation relationships in the CSOM model and the substance meronymy information as class attributes (because substances are probably not identifiable).

2.2 User-defined Relationships

Analogous to objects, the *meaning* of a certain relationship, denoted by a verb, is ambiguous. The right meaning should be chosen first, see Figure 2.4. The next step is to chose the *structure* or *occurrence* of the verb: how many objects are involved and *what kind* of objects are allowed in the relationship (Figure 2.4). The number of objects involved in a relationship is stored in WordNet as *verb frames*. Unfortunately, WordNet does only discriminate between *'somebody'* and *'something'*, so that the kind of object is not strongly restricted by using WordNet. This limited discrimination is enough, however, to prohibit relationships as *a bike borrows a book* because a *bike* is not a *somebody*. We have manually added a list that relates WordNets verb frames to full verb specifications containing semantic functions. This addition facilitates the generation of natural language sentences considerably.

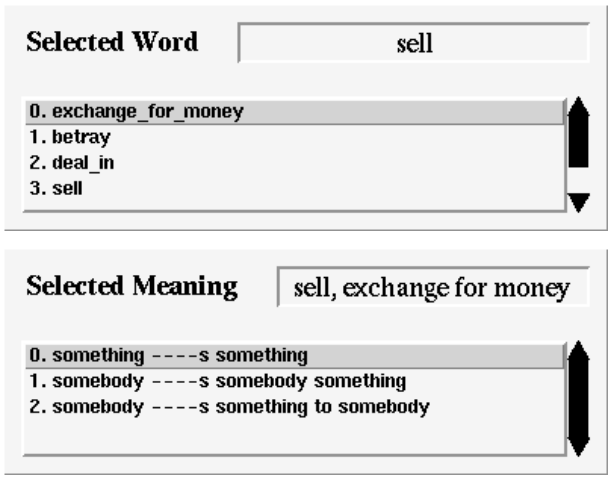

Figure 2.4: Specifying the meaning and occurrence of 'sell'

2.3 Standard Relationships

Both CSOM and WordNet contain standard relationships, and they are closely related. If we talk about standard relationships in this paper we mean standard *conceptual* relationships, as listed in chapter 1. The following information retrieved from WordNet points to a standard (conceptual) relationship:

- hyponymy/hypernymy : specialization/generalization (**is_a**)
- part and substance meronymy : aggregation (**has_a**)

The possession and instantation relationships are domain specific, and this information is not available in WordNet.

2.4 Other Knowledge

Among the other information we retrieve from WordNet are the *antonym*, *entailment* and *causal* relationships between verbs. This knowledge is used to make the models more complete, by offering additional information about the events that may occur in the Universe of Discourse.

Antonym Events Almost every current conceptual modeling method contains some step in which the events occurring in the universe of discourse are listed, see for example OMTs *event traces* [RBP+91] or Jacobsons *use cases* [Jac92]. CEMs do not only contain this kind of information, but also formalize it. Although the initial step, listing the events and ordering them in time in an informal way, should be done manually by the modeler, the creation of the CEM itself is embedded, and thus supported, by a CASE-environment. The availability of standard building blocks is already mentioned before. The event specifications are build up in the same way as the user-defined relationships of the CSOM model, because they are both CPL specifications that only differ in their tense (active and passive, respectively). This means that the determination of the meaning, occurrence and attributes of verbs and nouns is supported by WordNet. To end up with CEM models that are complete (to a certain degree), the *antonym* events of the events occurring in the requirements document are generated as well. Most of the times, these antonym events occur in the universe of discourse but are not always included in the requirements document explicitly. In the library-example, the *free*-event was generated as antonym of the *block*-event and added to the CEM. The antonym-event of *borrow* (i.e. *return*), however, is already appearing in the model. The problem we discovered using WordNet to find these antonyms, was that the antonym of *borrow* was not *return*, analoguous to the *block-free* antonym couple, but *lend*. The *borrow-lend* is a perspective (or conversive) antonomy (the subject and object are switched) and not a complementary (or reversive) antonomy. Both relationships are important, but the mixture of both into one relationship restricts their use.

Entailment and Cause-to Relationship The entailment and cause-to relationships give more detailed information about what the exact contents of a certain event are. For example, the selling of a car to a customer, does not only mean that the car is exchanged for money, but means also *transfering possession of the car (something concrete or abstract) to the customer (somebody).* This is a typical example of a dynamic effect on the static structure of an environment (the sell-event entails the change of the owner of the possession relationship between person and car). The presence of the entail relationship gives thus further insight in the meaning of events and the relations between events and static structures. The cause-to relationships is especially useful when relationships between events are investigated, e.g. to take care of possible side-effects of events.

2.5 Verification and Validation

Because of the fact that we have extended the conceptual model with linguistic features and that we are using WordNet as a lexicon, we have created more possibilities to validate and verify the intermediate and resulting models. Summarizing the syntactic and semantic verifications we have computed so far:

- right word category at right place in model
- kind of nouns connected to verb
- constraining the combinations of labels and primitive modeling concepts at a meta-model level
- relationships between nouns
- standard building blocks

- reusable information and (parts of) models from the reusable knowledge base
- consistency among specifications (i.e. parts of models)

These verifications have led to models that are syntactically and semantically correct (i.e. *the models are right*) but there is still the possibility that the models do not express what is described in the requirements document (i.e. *they are not the right models*). This could happen easily, by misunderstandings, misjudgements or bad communications, and should be recognized as being a problem. Our solution to this problem is the generation of natural language sentences, expressing the contents of the models. These paraphrases and the models themselves should be compared to the requirements document to see if they are indeed an abstraction of this document. We will summarize our natural language generation process, which is facilitated by the fact that we have based COLOR-X on CPL. CPL, which has existed for quite some years now, is based on Functional Grammar [Dik78] [Dik89], which makes the natural language generation rather straightforward. Although we think that this kind of natural language processing should be supported with a lexicon that contains lexical information about words but also rules for derivations, WordNet was not build explicitly to facilitate natural language generation. Although we access the WordNet files directly, and therefore not using WordNets morphological processing functions, we also believe that inflectional forms of words should be generated by *rules*. We have created a lexicon, that consists of WordNet augmented with new and optimized rules about verb inflections, plural and singular form of nouns, numerals, adjectives, determiners, etc. Our Prolog-translator CPL2NL translates any form of CPL-specifications into correct Natural Language sentences. We will discuss some aspects of the CPL speci-

fications that have their impact on the generated sentences. First, the modality determines the auxiliary verb of the sentence as follows: **NEC**, **MUST** and **PERMIT** trigger *obliged to*, *should* and *permitted to*, respectively. Secondly, the cardinality of the subject (agent or zero) of the relationship determines the singular (exactly one) or plural form (*n* (or more/less)) of the related verb. The identification of the objects, that gives some more detailed information about the nouns, is added as a subordinate clause, starting with *where*. Finally, the satellites of the CPL specification are translated into adjuncts of place or time.

There are three basic forms of CPL-specifications:

Note: The consonant sound of *user* is not noticed because we do not use a phonetical analyzer. Therefore, the article *an* is generated instead of *a*.

1. *Unconditional:*

   PERMIT:ACTION: borrow(ag=user)(<+>go=book)(<1>src=library)

   `[an,user,is,permitted to,borrow,one or more,books,from,a,library]`

2. *Conditional:*

   MUST:PROSP: return(ag=user)(<+>go=book)(<1>dest=library)
   (**sit:** **PERF:** borrow(ag=user)(<+>go=book)(<1>src=library))

   ```
   [if,an,user,borrowed,one or more,books,from,a,library,
    then,an,user,will have to,return,one or more,books,to,a,library]
   ```

3. *Identified:*

   PERF: borrow(ag=user)(go=book)(tmp=V1 in time)(id: V1 = yesterday)

   `[an,user,borrowed,a,book,at,a,time,V1,where,V1,is,yesterday]`

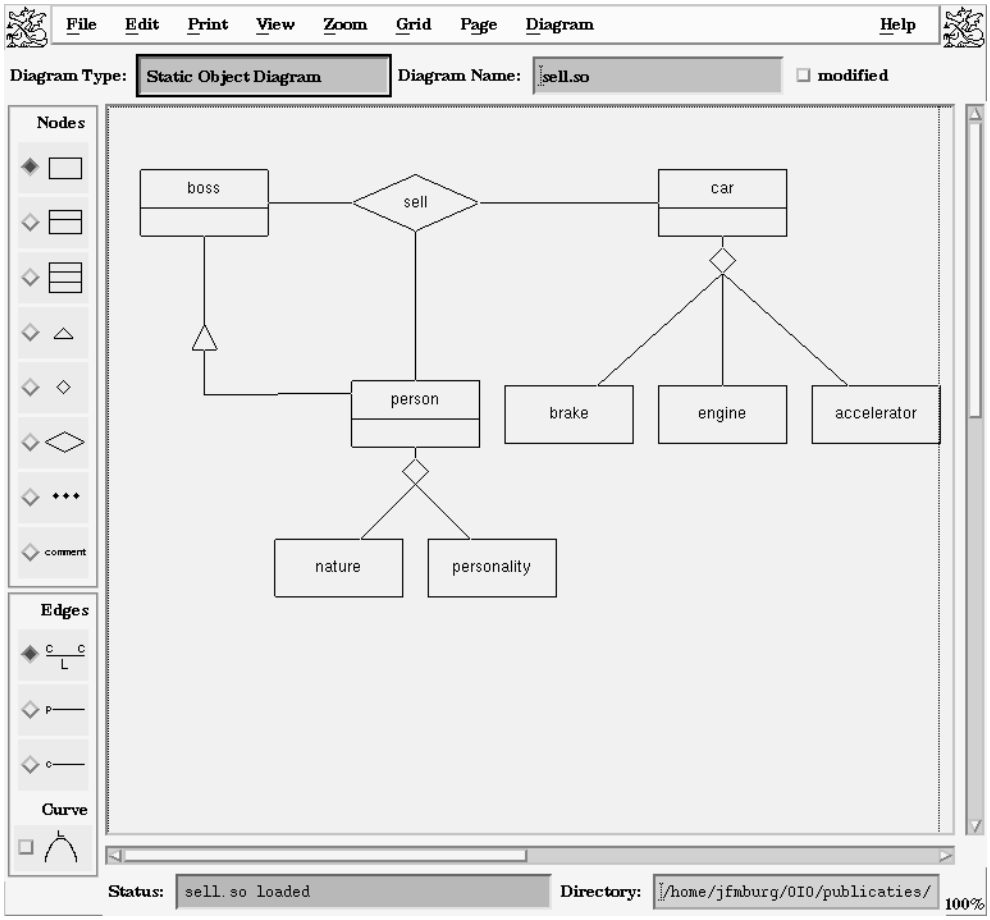

Figure 2.5: CSOM Model, generated from the WordNet consultations

2.6 Combining the Results

In the previous sections, we started with a single verb, and retrieved enough information from WordNet to build the corresponding CSOM completely, as Figure 2.5 shows. We will summarize the steps taken:

1. choose the name of relationship that one wants to model from the requirements document (for example: *'sell'*)

2. consult WordNet to disambiguate the meaning of the verb (for example: *'exchange for money'*) by examining synonyms

3. choose the syntactic usage (occurrence) of the verb from the set of verb frames stored in WordNet (*'somebody sells something to somebody'*)

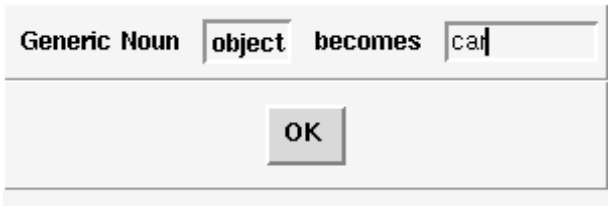

Figure 2.6: Specifying something to car

4. specify, if necessary, the elements of the verb frame (e.g. *'something'* becomes *'car'*, see Figure 2.6)

5. consult WordNet to find the right meaning for the nouns (e.g. *'car'* means automobile, as Figure 2.3 shows) by examining synonyms

6. if available, a list of possible attributes of the objects, found in WordNet is given. Choose among them (e.g. *'car'* has a *'engine'*) those that are relevant to the intended sense

7. there could exist some relationships, contained in WordNet, between two specified nouns (like *'boss'* is a *'person'*), that is not explicitly put in by the modeler. The opportunity is given to include these relationships into the final model

2.7 Alternative Approach

An alternative way of using a lexicon in the conceptual modeling process may be its incorporation into the requirements analysis phase. The approach we followed in the beginning of this section was to use the words retrieved from the requirements document in the analysis and design phase and find their meaning by consulting WordNet. In another small project we consulted WordNet during the requirements analysis phase, because:

1. it is easier to retrieve the right meaning from the lexicon because the words that are examined can be found in their original context

2. it is desirable to know as early as possible which concepts exactly are described by the words from the requirements document

We have built a small demo that gives the opportunity to:

- retrieve words denoting objects and relationships from the document

- find all the occurrences of these words

- find the correct meaning of these words in WordNet

Figure 2.7 shows the interface of this demo, including a part of the requirements document describing the library case, some retrieved objects and relationships, an alphabetically sorted list of all the words occurring in the document, and an interface with WordNet. The resemblance with tagging interfaces is clearly perceptible, as mentioned before.

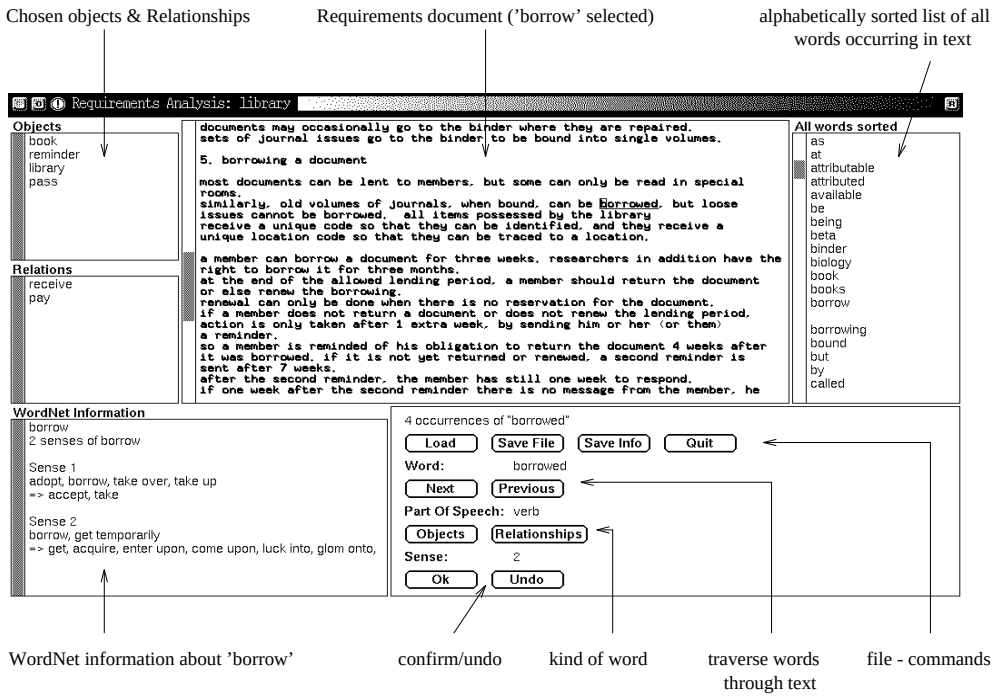

Figure 2.7: Using WordNet during Requirements Analysis

3 WordNet and a Reusable Knowledge Library

As we have shown in the previous section, we use WordNet as a source for reusable general information for building conceptual models. In fact, we use WordNet as a *base* for a much bigger, but more specific **reusable knowledge library**. The lexical knowledge from WordNet is applicable in every universe of discourse, but most of the times, the meaning of a certain word used in a universe of discourse is much more restricted.

We see the reusable knowledge library to be a multi-level repository, based on the specificity, or refinement, of the concepts found at each level. Most of the times the refinement levels would be:

1. Lexical Knowledge (WordNet) This kind of knowledge is rather static and will not change very easily, but can be extended with new concepts or new insights.

2. Domain-specific Knowledge Domain-specific knowledge consists of concepts and relationships between these concepts, which are typical for the domain (like a library-domain), but their specific meaning cannot be found in the lexical knowledge level.

This kind of knowledge should be built from scratch, but the lexical knowledge serves of course as a good starting point because the knowledge at this level will be derived from the lexical knowledge. The knowledge available will be used when a new application is built, but will also be extended if new knowledge is discovered during the construction of new applications that proves to be common for the domain or company as a whole. After some time, depending on the rate at which new applications are built and new definitions are elicited, the domain-specific knowledge will converge to a rather static collection. It is probably not a good idea to physically include this knowledge into WordNet, because it might be not valid in other domains.

3. Application-specific Knowledge Application-specific knowledge further refines the information available at the domain-specific level. The meaning of the concepts and relationships between the concepts found at this level, are not valid for the domain or company as a whole, but they are specific for the corresponding application, that uses this meaning.

This last level of knowledge consists of different sets, which are not necessarily disjoint, belonging to separate applications. The knowledge can be reused when the application is revised or a new version has to be built. Each set will be fairly static after the construction of the application is finished. The number of words with several meanings (homonyms and polysemes) should be minimalized to reach a certain level of standardization within a company. The amount of application-specific knowledge should therefore be as small as possible.

Because of the fact that all levels of knowledge tend to become rather static after some time, the information remains valid and serves as a good source for reuse.

An example, taken from the domain of the book-circulation desk of a library, will give the information available about a *book* for the three different levels:

lexical level: *a number of printed or written pages bound together* containing *text, chapters, foreword, contents, graphics, cover, etc.*

domain-specific level: *all books possessed by the library have a unique code so that they can be identified, and they have a unique location code so that they can be traced to a location. Books can be lent to members of the library*

application-specific level: *most books can be lent to members, but some can only be read in special rooms*, which leads to additional information about the lendability of books

3.1 Reusable Library: Models or Implementations?

In the software engineering field, the reusability issue is a very important one. However, some tendency in shifting the focus from reusable software components to reusable models [FP94], [CDAFP94] is visible and most of the time focuses on reusable *object* specifications. An important addition to this reusable knowledge is knowledge about relationships between objects and constraints on the objects and relationships, such as we have described in the previous section.

We have also carried out some research in the field of reusable software components and a supporting reusable software library [CEdJL93].

The incorporation of linguistic knowledge was a central theme. In particular the search facilities, indexed by identified nouns and verbs, could be supported with lexical background knowledge (descriptions of components could be parsed, synonyms of search items were found as well, etc.).

The way we used our Reusable Knowledge Library is however quite different from the traditional one. We did not search for models or software components directly, but for knowledge describing a certain object, relationship or event. If we found the desired piece of knowledge we had the possibility to incorporate it into conceptual models, or to translate it into descriptions of software components.

4 WordNet evaluated from a Conceptual Modeling Viewpoint

We have been using WordNet in a conceptual modeling environment for some time now, and although we still feel a need for additions or adjustments, we think it is a very powerful supporting tool for the process of conceptual modeling. While we have not said so earlier, the use of WordNet during conceptual modeling could take place after the models have been finished, or during the construction of the models. The latter possibility is the one used and described in this paper. The first one can be seen as verifying the created models, which could lead to a reconstruction of the models in case they are not right. This way of repeating the same steps (i.e. constructing a model), was exactly the reason that we have used the WordNet information *during* the construction process.

We will highlight some important strong points of WordNet used in our environment:

1. the large amount of words and concepts

2. the determination of the *meaning* of words

3. the 'common knowledge'-nature of the information about concepts, which frees the modeler from reinventing the wheel over and over again

4. the availability of verb frames, which prevents the modeler from leaving parts of relationships unspecified, or of specifying the wrong type

5. the open architecture, which makes it possible to access the information from every application

For our purpose of supporting the process of conceptual modeling, WordNet would even be more powerful and useful, if the following kinds of information (or knowledge) about words and concepts would be available:

1. *semantic functions* (or thematic roles), i.e. the exact roles the elements of a verb frame play. For example *'somebody sells something to somebody'* would be more powerful if it was stored as *'sell (agent=somebody) (goal=something) (recipient=somebody)'*. The semantic functions and related *satellites* are defined well in the theory of Functional Grammar [Dik78] [Dik89];

2. improvements in the verb frames also includes a more precise specification of the *type* the elements of the frame should have (i.e. females breastfeed babies). For WordNet to be as general as possible, this improvement would be difficult, because a more precise specification could restrict the applicability of the frame to only a few situations. Some sort of leveling, as we propose in our

three-level reusable knowledge library for example, could solve this problem;

3. characteristic information (*features*) about concepts (e.g. durative, controlled, telic, dynamic) which are especially useful in combination with the previous two points mentioned. To verify if some concepts could be combined, the features should match;

4. additional 'common knowledge' about concepts (for example: *'a person has a name'* is commonly known to be true, and is very useful for conceptual modeling). A comment has to be made about these proposed additions: it is very hard to determine what is commonly accepted and what is not. There is much disagreement among people on this question;

5. the presence of rules for determining derivations of words, i.e. inflectional and derivational morphology

6. the gender and [+human]/[-human] aspect of nouns, e.g. for determining relative pronouns during natural language generation;

7. a strict separation between perspective antonomy (e.g. borrow-lend) and complementary antonomy (e.g. borrow-return).

5 Conclusions and Further Research

As we have shown in this paper, the introduction of linguistic knowledge and theories into conceptual modeling is a viable one. The resulting models, mainly enforced by reusing information from WordNet and the related reusable knowledge library, do not only tend to be syntactically and semantically correct, but they are also consistent with the requirements documents, due to the generation of natural language sentences paraphrasing the models; they are moreover mutually consistent.

We have dealt succesfully with two important integration orientations in the software engineering life cycle, by using the same concepts for each phase, and by using WordNet information and linguistic theories to adjust the analysis and design phase (vertical integration), and by using the same underlying formalism for both the static and the dynamic models (horizontal integration). Issues that are still topics of research include *Rhetorical relationships* (like WordNets "entail" and "cause-to" relationships) [MT87]. These could be used to interrelate events with other events and states of the system as whole. The dependencies between natural language specifications in the requirements document and the code that is generated are also issues that have to be worked out (an example: the promise to do something in the future will have its consequences on the implemented code, which should include something as a list of such promises that trigger some events when they are not fulfilled in time). In the COLOR-X project we are mainly using natural language *generation* out of models, which increases the communicative properties of the models. The generated sentences are also used for validating the models, as we [BvdR94], [BvdR95b], [BvdR95a] and others [Dal92] have described. Natural language *parsing*, on the other hand, could be used for creating models [RP92]. The output of this parsing process could be an ER-model [TCY92] or a NIAM-like schema [Bla87]. The parsing of the requirements document could be a useful addition to the creation of COLOR-X models.

At the moment we are working on a case in which a system is being developed for planning and scheduling educational activities at the university. Although we are still in a preliminary phase, the results are promising.

References

[Bla87] W.J. Black. Acquisition of Conceptual Data Models from Natural Language Descriptions. In *Proceedings of the 2nd Conference of the European Chapter of the ACL*, Copenhagen, 1987.

[BvdR92a] P. Buitelaar and R.P. van de Riet. A Feasibility Study in Linguistically Motivated Object-Oriented Conceptual Design of Information Systems. Technical Report IR-293, Vrije Universiteit, Amsterdam, 1992.

[BvdR92b] P. Buitelaar and R.P. van de Riet. The Use of a Lexicon to interpret ER Diagrams: a LIKE Project. In G. Pernul and A.M. Tjoa, editors, *Proceedings of the 11th International Entity-Relationship Conference (ER'92)*, pages 162–177, Karlsruhe, 1992. Springer-Verlag.

[BvdR94] J.F.M. Burg and R.P. van de Riet. COLOR-X: Object Modeling profits from Linguistics. Technical Report IR-365, Vrije Universiteit, Amsterdam, 1994.

[BvdR95a] J.F.M. Burg and R.P. van de Riet. COLOR-X: Linguistically-based Event Modeling: A General Approach to Dynamic Modeling. In J. Iivari, K. Lyytinen, and M. Rossi, editors, *The Proceedings of the Seventh International Conference on Advanced Information System Engineering (CAiSE'95)*, number 932 in Lecture Notes in Computer Science, pages 26–39, Jyvaskyla, Finland, 1995. Springer-Verlag.

[BvdR95b] J.F.M. Burg and R.P. van de Riet. COLOR-X: Object Modeling profits from Linguistics. In N.J.I. Mars, editor, *Towards Very Large Knowledge Bases: Knowledge Building*

& Knowledge Sharing (KB&KS'95), pages 204–214, Enschede, The Netherlands, 1995. IOS Press, Amsterdam.

[BvdR95c] J.F.M. Burg and R.P. van de Riet. The Impact of Linguistics on Conceptual Models: Consistency and Understandability. In M. Bouzeghoub and E. Metais, editors, *First International Workshop on Applications of Natural Language to Data Bases (NLDB'95)*, pages 183–197, Versailles, France, 1995. AFCET.

[BvdRC93] J.F.M. Burg, R.P. van de Riet, and S.C. Chang. A data-dictionary as a lexicon: An application of linguistics in information systems. In B.Bhargava, T.Finin, and Y.Yesha, editors, *Proceedings of the 2nd International Conference on Information and Knowledge Management*, pages 114–123, 1993.

[CDAFP94] S. Castano, V. De Antonellis, C. Francalanci, and B. Pernici. A Reusability-Based Comparison of Requirement Specification Methodologies. In *Proceedings of the IFIP 8.1 Conference, CRIS*, Maastricht, 1994.

[CEdJL93] M.G.H. Coudron, J.A.M. Eerbeek, M.W.G. de Jong, and A.G. Leckie. Reusable Software Library: Design and Implementation. Master's thesis, Vrije Universiteit/GAK, Amsterdam, 1993.

[Dal92] H. Dalianis. A Method for Validating a Conceptual Model by Natural Language Discourse Generation. In P. Loucopoulos, editor, *Proceedings of the 4th International Conference on Advanced Information Systems Engineering (CAiSE'92)*, number 593 in Lecture Notes in Computer Science, pages 425–444, Manchester, UK, 1992. Springer-

Verlag.

[Dig89] F.P.M. Dignum. *A Language for Modelling Knowledge Bases. Based on Linguistics, Founded in Logic.* PhD thesis, Vrije Universiteit, Amsterdam, 1989.

[Dik78] S.C. Dik. *Functional Grammar.* Amsterdam, 1978.

[Dik89] S.C. Dik. *The Theory of Functional Grammar. Part I: The Structure of the Clause.* Floris Publications, Dordrecht, 1989.

[DvdR91] F.P.M. Dignum and R.P. van de Riet. How the modelling of knowledge bases can be based on linguistics and founded in logic. *Data and Knowledge Engineering Journal*, 7:1–34, 1991.

[FP94] C. Francalanci and B. Pernici. Abstraction Levels for Entity-Relationship Schemas. In *Proceedings of the 13th International Conference on the ER-approach*, Manchester, 1994. Springer Verlag.

[FW93] R.B. Feenstra and R.J. Wieringa. LCM 3.0: A Language for Describing Conceptual Models – Syntax Definition. Technical Report IR-344, Vrije Universiteit, Amsterdam, 1993.

[Gul93] J.A. Gulla. *Deep Explanation Generation in Conceptual Modeling Environments.* PhD thesis, University of Trondheim, Trondheim, 1993.

[Jac92] I. Jacobson. *Object-Oriented Software Engineering: A Use Case Approach.* Addison-Wesley, 1992.

[MBF+93] G.A. Miller, R. Beckwith, C. Fellbaum, D. Gross, K. Miller, and R. Tengi. Five Papers on WordNet. Technical report, Cognitive Science Laboratory, Princeton University, 1993.

[MT87] W.C. Mann and S.A. Thompson. Rhetorical Structure Theory: Description and Construction of Text Structures. In G. Kempen, editor, *Natural Language Generation: New Results in Artificial Intelligence, Psychology and Linguistics*, pages 85–95. Martinus Nijhoff Publishers, 1987.

[RBP+91] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, and W. Lorensen. *Object-Oriented Modeling and Design.* Prentice-Hall International, Inc., Englewood Cliffs, New Yersey, 1991.

[RP92] C. Rolland and C. Proix. A Natural Language Approach for Requirements Engineering. In P. Loucopoulos, editor, *Proceedings of the 4th International Conference on Advanced Information Systems Engineering.* Springer-Verlag, Manchester, 1992.

[TCY92] F.S.C. Tseng, A.L.P. Chen, and W-P. Yang. On Mapping Natural Language Constructs into Relational Algebra through E-R Representation. *Data and Knowledge Engineering*, 9:97–118, 1992.

[vdR94] R.P. van de Riet. Linguistic Instruments in Knowledge Engineering, A Research Proposal and some Experiments. In K. Fuchi and T. Yokoi, editors, *Knowledge Building and Knowledge Sharing*, pages 200–207. Ohmsha (Tokyo) and IOS Press (Amsterdam), 1994.

[vdVvdR94] A.J. van der Vos and R.P. van de Riet. A First Semantic Check based on Linguistic Information for State Transition Diagrams. Technical Report IR-372, Vrije Universiteit, Amsterdam, 1994.