ELSEVIER

Data & Knowledge Engineering 35 (2000) 111–120

DATA & KNOWLEDGE ENGINEERING

www.elsevier.com/locate/datak

Linguistically based requirements engineering – The NIBA-project

Günther Fliedl [a,*], Christian Kop [b], Heinrich C. Mayr [b], Willi Mayerthaler [a], Christian Winkler [a]

[a] *Institute for Linguistics and Computational Linguistics, University of Klagenfurt, Universitatsstrasse 65-67, 9020 Klagenfurt, Germany*

[b] *Institute for Business Informatics and Information Systems, University of Klagenfurt, Klagenfurt, Germany*

Received 31 January 2000; received in revised form 31 January 2000; accepted 25 May 2000

Abstract

Usually, the development of an information system (or some part of it) starts with requirement elicitation followed by a phase of collecting and analyzing which results in a set of requirements specifications. As opposed to conventional conceptual modeling, where input texts are formalized, our approach suggests the idea of collecting and cataloguing natural language patterns in a non-textual form immediately after a linguistic analysis. This linguistic analysis is done according to the NTMS model. Collecting and cataloguing of natural language data is supported by KCPM. © 2000 Elsevier Science B.V. All rights reserved.

Keywords: Object oriented and conceptual modeling; Natural language processing; Requirements engineering; Syntactic and semantic parsing

1. Introduction

The analysis of requirement specifications in the context of information systems development is of growing importance in the field of computational linguistics.

Usually, the development of an information system (or some part of it) starts with requirements elicitation followed by a phase of collecting and analyzing which results in a set of natural language requirements specifications. The latter serve as a source for conceptual design, where a semantic model of the given universe of discourse is established using modeling and representation concepts like UML [1] or former approaches like OMT [8] or even the classical Entity–Relationship Model (or one of its numerous extensions). These approaches are thought for enhancing the validation of the collected requirements as well as for providing a more formal basis for the subsequent system design and realization steps. As a matter of fact, such approaches form

[*] Corresponding author.

E-mail addresses: guenther.fliedl@uni-klu.ac.at (G. Fliedl), chris@ifit.uni-klu.ac.at (C. Kop), mayr@ifit.uni-klu.ac.at (H. C. Mayr), willi.mayerthaler@uni-klu.ac.at (W. Mayerthaler), christian.winkler@uni-klu.ac.at (C. Winkler).

PII: S 0 1 6 9 - 0 2 3 X (0 0) 0 0 0 2 9 - X

a considerable help for the designer. On the other hand, they are not very helpful with respect to collecting and cataloguing information. Moreover, these approaches require a level of abstraction end-users which (i.e., the 'requirement suppliers') is often not willing or not able to cope with.

To overcome these disadvantages the Klagenfurt Conceptual Predesign Model (KCPM) was developed. KCPM is intended to harmonize the developer's and the end-user's view of a given universe of discourse (UoD), thus providing an interface for their mutual understanding. As opposed to conventional conceptual modeling, where input texts are formalized, KCPM suggests the idea of collecting and cataloguing UoD information in a non-textual form immediately after a linguistic analysis. This linguistic analysis is done according to the NTMS model, which stands for 'Natürlichkeitstheoretische Morphosyntax' (Naturalness Theoretical MorphoSyntax) [3,4,6]. NTMS is a generative orientated grammar model that uses category-specific phrase structures in the description of grammatical phenomena. In what follows we outline the main concepts of KCPM and some steps of the linguistic analysis, including a brief description of the NTMS-parser [5].

2. KCPM

The language for KCPM consists of a set of simple notions we are going to explain in this section. Roughly speaking, every language dealing with information system analysis has to handle at least the following three aspects:

- structure (static part);
- functionality;
- behavior.

In order to provide a language that enhances the communication between developers and end-users we concentrated on a small number of modeling concepts (notions), namely *thing type*, *connection type*, *operation type*, *event type* and *constraint*. These are the main concepts other notions are derived from.

In the subsequent sections we will present the static part of these notions which are important during requirements analysis. We mainly discuss how the different notions relate to each other in Fig. 1. This is why we do not indicate the attributes of the meta classes in this figure, but mention only the most important ones.

2.1. The static part

The notions *thing type* and *connection type* are intended to cover static UoD aspects. Consider *thing type* as a generalization of the well-known conceptual notions *entity-type* resp. *class* and *attribute* resp. *value type*. Thus, *thing type* covers general concepts like natural or juridical persons (e.g. *customer*, *employee*), material or immaterial objects (e.g. *product*, *vehicle*, *account*, *order*), features (e.g. *customer name*, *article number*, *color*, *weight*). Beyond that, a *thing type* represents an essential concept in the UoD. These concepts are frequently expressed by noun phrases. Linguistically speaking, a *thing type* is a *noun phrase* represented by either a simple noun (e.g. *customer*) or a complex noun phrase (e.g. *ordering a product*).

The fact that stake holders often use synonyms and simply provide examples is accounted for by means of the association *is_synonym_of* and the meta-class *example*. The association *is_synonym_of* means that there are several synonyms for one *thing type*, however, there is only one

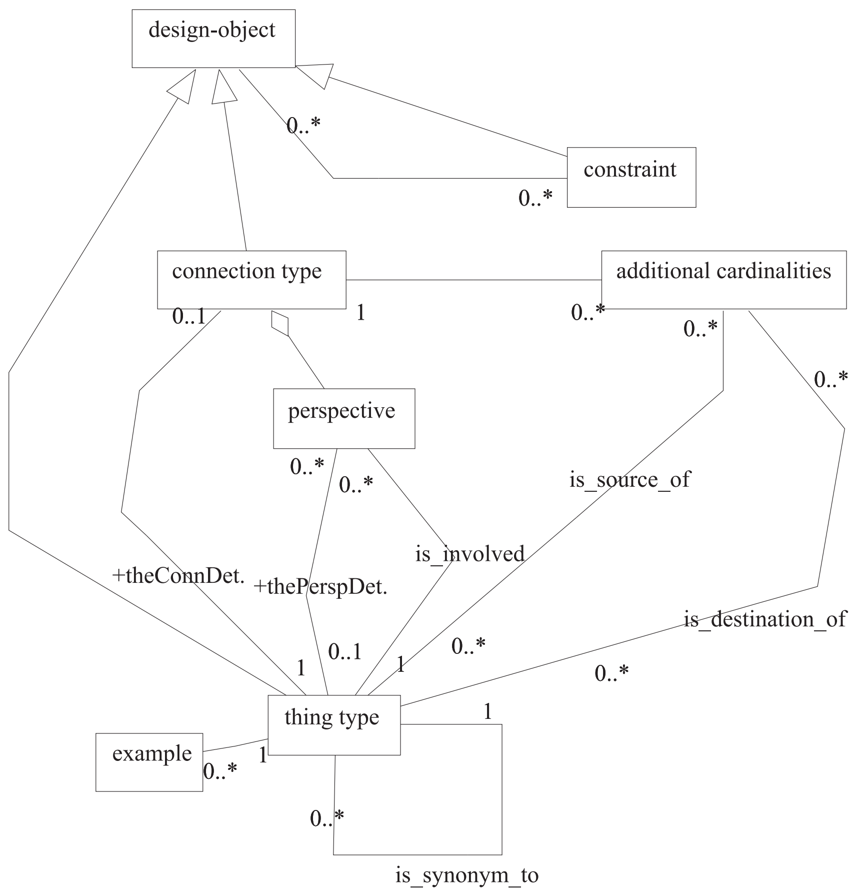


Fig. 1. Static part.

thing type we will work with. An important notion not shown in the graphical representation of our language is *quantity description*. Each entry of a *thing type* is provided with a column for quantity description where we can store information like *we have 1000 customers* or *per year we get 10 customers more* etc. Such information might be of considerable importance with respect to tuning databases. On the other hand, it can be elicited easily during requirement analysis.

Not distinguishing between *classes* and *value types* implies not distinguishing between relationships (e.g. associations and attributes) either. KCPM proposes the concept *connection type* for all kinds of relationships including the abstraction *aggregation* and *generalization*. The following sentences relate several *thing types*, more precisely *customer* with *product* and *customer* with *phone number*, thus building up a *connection type*:

1. *A customer can buy more than one product* (*Ein Kunde kann mehrere Produkte kaufen.*)
2. *A product is sold to exactly one customer* (*Ein Produkt wird an genau einen Kunden verkauft.*)
3. *A customer has up to four phone numbers* (*Ein Kunde hat bis zu vier Telefonnummern.*)
4. *A phone number belongs to one customer* (*Eine Telefonnummer gehört zu einem Kunden.*)

Defining a *connection type* exhaustively requires its description from the *perspective* (point of view) of all the *thing types* involved.

Note that this approach corresponds to the NIAM Object-Role Model [7]. (1) and (2) are *perspectives* of one *connection type*. Furthermore, (3) and (4) can be considered as *perspectives* which together build up a *connection type*. Quantities of a connection (*cardinalities*), e.g. *a maximum of 4 phone numbers* are specified within a *perspective* (not shown in the graphical representation of the language).

The abstractions *generalization* ('is-a', with set inclusion on the instance level) e.g. *a truck is a car* and *component/object* ('is part of') are treated as specific *connection types*. Furthermore, an *identification connection type* can be defined. Due to the following reasons the model has to be open for further semantic *connection types* (e.g. *possession*). First, we want to transform the entries not only to an object-oriented model, but also to the entity relationship model, which needs such a connection. Second, in typical UoDs such connections do exist, think e.g. of an identifying *customer number* for a customer. Since such identifying *thing-types* are always part of a universe of discourse and easy to find, the assumption of an *identification connection type* with the perspectives *identifies* and *identified_by* (e.g. *a customer-number identifies a customer*) is justified. Thus, you can simply define special semantic *connection types* by assigning each perspective a defined label.

To capture the situation that a *connection type* itself is an essential UoD notion, expressed by a noun phrase, a separate association between *connection type* and *thing type* has been established. In this association *thing type* plays the role of a *connection type determiner*. Moreover, a *thing type* can play the role of a *perspective determiner*, which means, that a *perspective* itself is an UoD-notion labeled with a noun phrase. All other UoD-aspects that are not captured by the beforementioned KCPM concepts are called *constraints*.

2.2. The environment

KCPM does not enforce a specific representation. Instead of this we propose to employ glossaries, thus underlining the idea of using a KCPM-Schema as a kind of notepad during requirements collection. These schemata may also be understood as a dictionary for the given application domain.

In order to support this idea, the core model has to be completed by an environmental model expressing the relationship between *requirements* and *design objects*. Each design object is related to one or more *requirement sources* (and vice versa). In the simplest case a requirement source is represented by exactly one sentence. It goes without saying that any other part of a text document (e.g. a paragraph) may be analyzed as requirement source as well. Since we do not restrict to text documents, a variety of (additional) information such as images, graphics etc. can be regarded as requirements source. Each requirement source has an *informant* who stated the requirement. For each design object the designer can note remarks or open questions. Both are *memos* where the last type has to be answered by a specific informant.

Each design object is embedded into a working environment. The basic concept of the working environment is the *Universe of Discourse*. If the UoD is very large, it has to be split into *organizational environments* such as departments of an enterprise or fields of activities.

To sum up, our approach is called 'conceptual' since it employs semantic modeling concepts; it is called 'predesign' since it forms a very early modeling step within the information systems design process. Of course, using KCPM would be very time consuming if there were no tools supporting the collection and classification of linguistic objects and no mechanisms for automatizing the transformation of a KCPM-Schema into a conceptual schema (e.g. a UML model) either. This is why the NIBA-Project [1] aims at providing such tools based on the related theory.

3. Linguistic support for the extraction of KCPM notions

Knowledge processing as required in conceptual predesign needs a highly efficient computational linguistic model for the analysis of language. The aim of the NTMS approach is to comply with such requirements with regard to praxis oriented computational linguistic work. In the following, some features of the model:

- lexicon entries are classified by features and organized corresponding to application domains;
- semantic rules;
- grammatical values (e.g. subject function) and sentence context (e.g. argument structure) are determining parameters; these are made explicit according to naturalness theory.

Computational grammar is considered as the *missing link* between the UoD in question and the schemata applied/defined using KCPM. In other words, semantically enriched syntactic structures can be translated directly into KCPM modeling concepts, the input being any German sentence related to the chosen segment of reality. The analysis contains the following main steps:

- the words of a given text are compiled into a lexicon which implies automatic categorial, semantic and contextual specification of words;
- word and morphosyntactically interpreted microstructures are assigned to wordgroups and words, e.g. the automatic splitting of German compound nouns [3];
- analysis of (sentence) syntax according to the specific X'-mechanism of NTMS [6]. This generates binary branching trees or labeled bracketings with enriched semantic features for phrases;
- bracketing output is listed and compared; the analyzed sentences are listed and sorted with respect to their textual information and position in the text produced by the end-user.

The bracketing-output contains the semantic features of the analysis step and may be used for the interpretation of the sentence itself and for the extraction of KCPM concepts. Generally speaking, in this step we extract *thing types* from noun phrases (n3; without the need to distinguish whether the noun phrase is a class or an attribute). From verb–phrases we extract connection types or operations respectively. The semantic roles (⟨AG;TH⟩; a specific generated semantic feature) a noun phrase embraces within the verb–phrase indicates its specific function in an operation and a connection type, e.g.

Ein Kunde kann mehrere Produkte kaufen

v4(n3(n2(q0([Ein]),n0([Kunde_k]))),v3(spz0([kann_mod-i]),v2(n3(n0([0_k])),
v1(n3(n2(q0([mehrere]),n0([Produkte_j]))),v0([kaufen⟨AG;TH⟩_i])))))

4. The NTMS-parser

The NTMS Parser recurs both on sentence internal relations between terminal elements and on sentence internal and sentence external morphosyntactic, semantic and discourse pragmatic

[1] NIBA is a German acronym for Natürlichsprachliche Informationsbedarfsanalyse (natural language requirement analysis). The project is sponsored by the Klaus Tschira Stiftung, Heidelberg.

features. As for memory saving reasons the linguistic system in language processing is divided into the following subsystems:

- The stem lexicon; not varying words, i.e. words, whose stems are not subject to word formation processes can be stored as stems. The NTMS-Parser contains verbal stem entries and the respective forms with predicate-argument-structure (PAS) in the indicative mode.
- The morphological component. It assigns affixes to potential stems and generates derived forms on the base of acceptibility conditions.
- The syntax algorithm is enriched with morphosyntactic features establishing agreement conditions for a great part of natural sentence patterns of standard German. On the base of this set of rules the parser checks whether an input sentence can be considered as syntactically, morphosyntactically and partly sentence-semantically correct and then generates labeled bracketing structures or tree diagrams in the postscript format.

Briefly, the main task of the NTMS Parser is the generation of a constituent structure enriched with semanto-pragmatic features for morphologically correct word chains, thus facilitating further processing in the field of text grammar and/or static and dynamic modeling.

Language engineering requires that a parser be capable of distinguishing between different domain specific meanings of sentences and of assigning several unequivocal structures to the respective phrases. The domain-semantic and discourse-pragmatic knowledge establishing the base of the 'intuition' of sentence meaning has to be specified adequately. NTMS specification of such knowledge is done on different levels, as discussed in the following section.

4.1. Levels of analysis

4.1.1. Lexical specification

Lexicon data have been collected in a relational data base [2] and transformed into class-specific and morphosyntactically relevant replacement rules with the help of VBA [3] functions. Up to now, the following steps have been completed:

- assigning the stems of simplex verbs to 20 semantically interpreted verb classes;
- generating verb class specific terminal rules for the morphosyntax parser;
- building up a nominal data base in which nouns are specified with regard to primary features such as [α animated] and to discourse pragmatic features such as [α definite].

The verb lexicon contains more than 5000 verb stems of the most frequent German simplex verbs, prefix verbs and particle verbs and the respective inflected forms such as follows:

1st person-indicative-present-singular;	1st person-indicative-present-plural;
2nd person-indicative-present-singular;	2nd person-indicative-present-plural;
3rd person-indicative-present-singular;	3rd person-indicative-present-plural;

[2] The lexicon and the concept component have been implemented in MS-ACCESS.
[3] VBA – Visual Basic for Applications.

the infinitive without *zu* (to) and the infinitive with *zu*; the past participle, morphologically corresponding to the passive participle.

As a whole, the lexicon contains roughly 50,000 entries of finite and non-finite German verb forms interpreted according to their predicate-argument-structure (PAS). Each stem form and subsequently the respective inflected forms carry at least one unmarked, at the most one unmarked and one marked (less common) PAS.

Class specific PAS are assigned to the verbs unequivocally, 12 general PAS corresponding to 12 numbered verb classes. The partly necessary subclasses (7,1; 7,2; 8,1; 8,2; etc.) have been established with regard to the different morphosyntactic features of the relevant verbs and their valency partners. With the help of this inventory of description of PAS and on the base of the considerations mentioned above German verbs can be classified as follows:

Class	**Abbreviation**[a]	**PAS**
1	AUX	$\langle\emptyset\rangle/V_{\text{fin}}$
2	eV	[TH]
3	iV	AG/TH[ ]
4	locV	TH[LOC]
5	possV	GO[TH]
6	psychV	TH[GO/TH]
7	tVag/2	AG[TH]
7,1	tVag/2d,g	AG[GO/BEN]
7,2	tVag/2pp[b]	AG[SO/GO]
7,3	tVag/2sk	AG[THabstr]
7,4	tV/ppsk	AG[GO/SOabstr]
8	tV/3	AG/GO(TH, GO/SO)
8,1	tV/3ti	AG[TH..., GOabstr]
8,2	tV/3tda	AG[THacc, GOabstr]
8,3	tV/3tdd	AG[THdat, GOabstr]
8,4	tV/3sk,ak	AG[GO, THabstr]
9	sentV	EXP[TH]
10	Vcop	TH[N2/A2]
11	tV/2	-AG/-EXP[TH]
12	reflV	AGi/THi[i, (GO/LOC/TH)]

[a] (1) AUX – auxiliary; (2) eV – ergative verb; (3) iV – intransitive verb; (4) locV – locative verb; (5) possV – possessive verb; (6) psychV – psychological verb; (7) tVag2 - bivalent agentverb; (7,1) tVag/2d, g – bivalent agentverb with dative-object or genitive-object; (7,2) tVag/2pp – bivalent agentverb with prepositional objekt; (7,3) tVag/2sk – bivalent agentverb with sentential object; (7,4) tVag/2pp – bivalent agentverb with sentential prepositional objekt; (8) tV/3 – trivalent verb; (8,1) tV/3ti – trivalent verb with infinitival complement sentence and thematical identity of the antecedent and the logical subjects of the infinitival group; (8,2) tV/3tdd – trivalent verb with infinitival complement sentence and thematical difference of the antecedent and the logical subject of the infinitival group; (8,3) (tV3tdd) – agentive, trivalent verb with concrete theme in the dative and abstract “GOAL”; (8,4) (tV/3sk,ak) – trivalent verb with infinitival complement and subject control or arbitrary control; (9) sentV – verba sentiendi; (10) Vcop – copula verb; (11) tV/2 – transitive verb, whose subject does not carry the AG-Role nor the EXP-role trägt; (12) reflV – reflexive verb.

[b] As for transparency reasons we note ‘pp’ instead of ‘p^2’.

Applying the above sketched input we then generate a terminal rule for the syntax parser by means of a second VBA-function. This rule is in fact a PROLOG DCG [4]-rule; compare the following example of a perception verb (sentV; class 9):

spz0(9,ps3,praes,ind,_,sing,_, [5]spz0([sieht_v_i]))->[sieht].

This rule defines *sieht* (sees) as V^0 under SPZ^0, thus assigning the finite verb the category SPZ^0 which in the NTMS model is reserved for the verb-second-position in German. The features assigned to the verb are the morphosyntactic conditions of unification allowing an acceptable sentence structure. The index '_v' with the verb symbolizes the full-verb character of *sieht*, the coreference index '_i' establishes the relationship with the end position of the verb.

The noun lexicon comprises a list of unmarked, frequent nouns and the respective desinences. It is continually updated and enlarged. Nouns have been subcategorized with respect to the parameters α agent, α animated, α derived, α compound etc.

Moreover, nouns have been labeled domain specifically and assigned to the domains trade, traffic, education and tourism. Within these selected domains homonyms, synonyms, hypernyms and hyperonyms have been worked out.

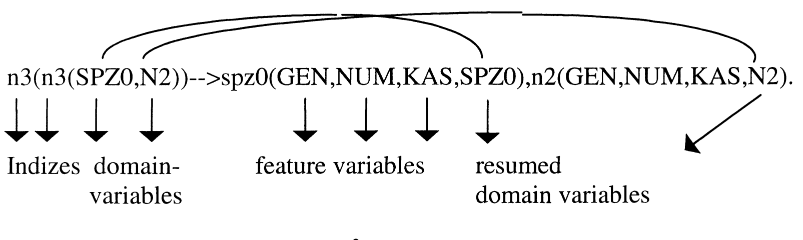
4.1.2. Syntactic specification in the X'-system

Chomsky [2] has suggested phrase structure rules generating not only phrasal categories (maximal projections) and terminal categories (minimal projections) but also intermediate categories. This is commonly known as X'-Theory. For further information on the specific X'-mechanism in the NTMS model see [3,6].

4.1.3. Morphosyntactic specification by means of enriched replacement rules

The DCG rules comprise constants which are either represented by indices/brackets or by features of parameters. Such bracketing structures contain variables (morphosyntactic parameters) that are replaced by features in further rules.

For instance, the NTS rule $\mathbf{N^3 \rightarrow SPZ^0\ N^2}$ is transformed in the following PROLOG [6]-structure:



```
n3(n3(SPZ0,N2))-->spz0(GEN,NUM,KAS,SPZ0),n2(GEN,NUM,KAS,N2).
```

This rule splits up the N^3-node, representing the maximal unfolding of a noun, into the two domains SPZ^0 and N^2. In addition to that, it has to be fixed how many parameters relevant for agreement are necessary for generating all possible word combinations and filtering out inacceptable structures.

[4] 'DCG' – definite clause grammar.

[5] '9'ps3, praes, ind,_,sing,_,' = Class 9; 3rd.person; present; indicative; unspecified domain; singular; unspecified cardinality.

[6] The grammar has been implemented in IF/Prolog (see Section 3).

4.2. Some features of the system

The NTMS-Parser has mainly been built to tackle the complex problems of German syntax in accordance with the respective domains. Thus, the tool has to be capable of:
- generating and indexing empty elements;
- illustrating discontinuous phrases (e.g. the so-called German 'verb-bracket');
- recognizing and illustrating syntactically motivated ambiguities;
- splitting up lexical elements with complex internal word structure if required by the respective application or domain (= partial word syntactic analysis, integrated in the sentence analysis);
- generating contextually dependent semanto-pragmatic features in the field of a text-grammar.

5. Conclusion

As opposed to conventional conceptual modeling KCPM suggests the idea of collecting and cataloguing UoD informations in a non-textual form immediately after a linguistic analysis in the NTMS model.

NTMS as a computational grammar is considered as the 'missing link' between an UoD and the schemata resulting from the use of KCPM. We gave a short sketch on how semantically enriched syntactic structures might be translated into KCPM modeling concepts, the input being any German sentence related to the chosen segments of reality. For this purpose, the process of requirements translation has been divided:
- into the NTMS-based semanto-syntactic analysis of natural language requirements specifications and
- the extraction of discourse semantics into a UoD glossary. We propose that these two subtasks are part of the first step of conceptual modeling, which we call conceptual predesign.

The next steps of our work aims at the extraction of dynamic UoD aspects and their mapping to KCPM concepts for dynamics and the before-mentioned transformation of KCPM schemes into UML models.

References

[1] G. Booch, J. Rumbaugh, I. Jacobson, The Unified Modeling Language User Guide, Addison Wesley, Reading, MA, 1999.

[2] N. Chomsky, Remarks on nominalization readings in english transformational grammar, in: Jacobs, Rosenbaum (Eds.), Waltham, Massachusetts, 1970.

[3] G. Fliedl, Natürlichkeitstheoretische Morphosyntax, Aspekte der Theorie und Implementierung, Habilitationsschrift, Gunter Narr Verlag, Tübingen, 1999.

[4] G. Fliedl,C. Kop, W. Mayerthaler, H.C. Mayr, C. Winkler, NTS-based derivation of KCPM perspective determiners, in: Proceedings of the Third International Workshop on Applications of Natural Language to Information Systems, 26–27 June 1997, The Harbour Centre Campus of SFU, 215-226. Vancouver, Simon Fraser University, 1997.

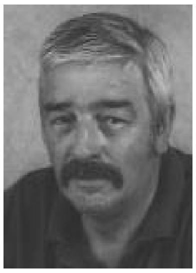
[5] G. Fliedl, W. Mayerthaler, C. Winkler, The NT(M)S Parser: an efficient computational linguistic tool, in: Proceedings of the First International Workshop on Computer Science and Information Technologies, 18–22 January 1999, Moscow, pp. 125–128.

[6] W. Mayerthaler, G. Fliedl, C. Winkler, Lexikon der Natürlichkeitstheoretischen Syntax und Morphosyntax, Stauffenburg Verlag, Tübingen, 1998.

[7] G.M. Nijssen, T.A. Halpin, Conceptual Schema and Relational Database Design – A fact oriented approach, Prentice Hall, Englewood Cliffs, NJ, 1989.

[8] J. Rumbaugh, M. Blaha, W. Premelani, F. Eddy, W. Lorensen, Object Oriented Modeling and Design, Prentice Hall, Englewood Cliffs, NJ, 1991.

Heinrich C. Mayr received his doctorate in applied mathematics from the University of Grenoble (France) in 1975. Between 1975 and 1983 he was an assistant professor at the university of Karlsruhe, Germany and a lecturer and visiting professor at several other German universities in the domain of database technology and information systems. From 1984–1990 he was CEO of a German software company, responsible for the section 'Business Information Systems'. Since 1990 he is full professor of informatics at the University of Klagenfurt, Austria. His current research includes information systems design methodologies, natural language processing in requirements analysis, case based reasoning in the context of help desk systems, methods and tools for production simulation in SME's, software project management and distance education. He is a member of ACM, GI, IEEE and OCG and others. Currently, he is the president of the German Informatics society (GI) and a member of the board of the Austrian Computer society (OCG).



Willi Mayerthaler studied Romance Languages, General Linguistics and Philosophy (focussed on Logics and Theory of Sciences) mainly at the Ludwigs-Maximilian-University in Munich, Bavaria, but also in Strasbourg (France), Oviedo (Spain), Lisboa (Portugal). He was an assistant professor at the Technische Universität Berlin for five years and became a full-time professor at the Universität Klagenfurt in 1979. Mayerthaler is co-founder and editor of the journal Papiere zur Linguistik (Tübingen, Narr). His main interests are Typology, Syntax-theory, Morphosyntax and Semantics. Together with Günther Fliedl, Mayerthaler is the mentor of the NT(M)S (Naturalness Theoretical (Morpho)syntax)). Last but not least, Mayerthaler plunged into Computational Linguistics with the special interest Fuzzy-theory.

Günther Fliedl first studied Philosophy and Music Science at the Universität Innsbruck. After coming to Klagenfurt University he switched to the combination Philosophy and General Linguistics. Fliedl has been teaching both General/Applied Linguistics and Computational Linguistics since1985 and has been involved in a variety of projects ever since. His main interests are Syntax and Morphosyntax, (syntactic and semantic) Parsing. Fliedl's post graduate studies brought to light the NT(M)S, a main component of the NIBA-Project.

Christian Winkler studied Romance Languages, English and General/Applied Linguistics at the Universität Klagenfurt. He has been teaching in many fields since 1994. Together with Willi Mayerthaler and Günther Fliedl, Winkler worked in quite a few projects and was also involved in the resulting publications. Main interests: Comparative Linguistics (comprising Phonetics, Phonology, Morphology, Syntax, Semantics), Computational Linguistics. Currently, the former clerk brings his experience into the NIBA Project.

Christian Kop studied Applied Computer Science at the University of Klagenfurt. He works as a research assistant at the Department of Business Informatics and Information Systems at this University. His research interests includes information systems analysis and design, and questions of natural language processing support for information systems analysis.