Conceptual Predesign
Bridging the Gap between Requirements and Conceptual Design

Christian Kop Heinrich C. Mayr
Dept. of Applied Informatics and Information Systems
University of Klagenfurt
A-9020 Klagenfurt, Austria
chris@ifi.uni-klu.ac.at mayr@ifi.uni-klu.ac.at

Abstract

To overcome the impedance mismatch between requirement analysis and conceptual design, we introduce an intermediate step between the two phases, called conceptual pre-design. A (semantic) model for that phase should allow for an easy collection of requirements as well as for an unproblematic transformation of the collected requirements into entries of a conceptual scheme. We present a model that has been developed along these postulates. Following the classical DATA-ID approach, this model uses a „glossary metaphor" for scheme representation. It's basic semantic notions are 'thing type' and 'connection type'.

1. Introduction

The information systems community usually divides the information system life cycle into 5 phases: requirements analysis, conceptual design, logical design, implementation/physical design and production/ maintenance. Methodical approaches to the early phases like methods of Object Oriented Analysis (OOA, see e.g. [CoYo91, RBPE91]) tend to integrate requirements analysis and conceptual design. As a consequence, the 'information sources', i.e. the future end-users, are confronted with the rather abstract conceptual modeling notions that are common to OOA approaches. Practical experiences show that this often leads to a design that is insufficiently transparent for the end-users. But transparency is necessary for validation purposes.

Moreover, similar to the classical ER-based conceptual modeling approaches OOA methods force the designer to early design decisions on what specific modeling notion has to be associated with a given aspect of the universe of discourse (UoD) in question. E.g., one has to distinguish between formal concepts like classes (objects), association types (relationships), attribute and value types. In contrast to that the main objective of requirements collection and analysis should be to „capture", as completely as possible, the relevant aspects of the given UoD.
Finally, if in a later analysis step such formal decisions turn out to be inadequate, changes with more or less complex local and global effects will become necessary. These changes reduce the design traceability and thus again impede the design validation by the end-users.

To overcome these problems we propose, in accordance to [Ceri83], to introduce between the phases of requirements analysis and conceptual design an intermediate phase which we call conceptual pre-design: Within this phase UoD relevant information is collected mainly in natural language form and „translated" by the means of a rather lean semantic model (the Klagenfurt Conceptual Pre-design Model KCPM) into a glossary-like pre-design scheme. The pre-design scheme then works as a detailed scratch pad for the given UoD.

Taking this approach we try to reach the following two goals:

- to automate, as far as possible, the process of producing the pre-design scheme by extracting its entries from the end-user's natural language requirements statements. This is investigated in NIBA[1,2], a project that we are running together with colleagues from computer linguistics.
- to automate, as far as possible, the mapping from conceptual pre-design to conceptual design.

KCPM offers a set of semantic concepts for modeling static and dynamic UoD aspects. It is based on an ontologic approach viewing UoD's as systems consisting of interrelated elements (things) that are able to perform services (operations) and that are activated by events (messages sent by other things). Thus, there is a strong relationship to object-oriented approaches that allows for a rather natural way to map KCPM concepts to OOA con-

[1] The project is financed partly by the Klaus Tschira Stiftung Heidelberg.
[2] NIBA is an acronym for **N**atürlichsprachliche **I**nformations**B**edarfs-**A**nalyse (natural language requirements analysis).

cepts. As will be seen within this paper, this leads, among others, to a systematic method for designing (deriving) class structures.

Within the present paper we will concentrate on the pre-design concepts for static UoD aspects and we will show how to map a static conceptual pre-design model to a conceptual one (the latter based on OMT [RBPE91]). The KCPM concepts for dynamic UoD aspects and their mapping to OOA (conceptual) models are actually investigated and will be published in a successor paper.

Section 2 gives an overview of related work and the position of our framework. In section 3 we describe the KCPM notions for modeling static UoD aspects and we present a meta scheme of our framework. Section 4 gives an overview of the NIBA project and reports on some practical experiences using KCPM. Section 5 presents the mapping steps to the conceptual scheme. The paper closes with a conclusion and some hints on future work.

2. Related Work

At the beginning of the eighties, several attempts have been made to develop rules, methods or tools to shorten the gap and to provide for a traceability between requirements (expressed in natural language sentences) and conceptual schemes [Eick84, KWDB86, Roll87]. Chen for instance introduced in [Chen83] eleven rules for transforming structured English sentences into ER-scheme entries.

The DATA-ID [Ceri83] approach uses glossaries as a means for requirements representation and proposes a process model for establishing these glossaries for a given UoD: In a 1st step, the UoD is to be split up into smaller areas (homogenous w.r.t. domain language used and/or w.r.t. common tasks). The 2nd step is devoted to the collection of 'requirement documents' (in any form) for each area. As far as these documents consist of natural language texts these then are simplified in order to enhance the extraction of semantics. The simplification (3rd step) is done using a set of syntactical and heuristic rules. Within the 4th step the sentences are classified by their key meaning, i.e. describing data, an operation or an event. Corresponding to the sentence class, within the 5th step information is entered into the glossaries that again are distinguished into data, operation and event glossary.

Another important approach is the conceptual model NIAM [NiHa89]. In contrast to other conceptual models, NIAM does not distinguish between entity types and attributes but introduces so-called *object types* that are connected to each other through *roles*. NIAM schemes are constructed by analyzing natural language sentences. Consider, e.g., the sentence „*the student with the name Smith studies the lecture with the title Software-Technology*": It allows for the derivation of the object types *student*, *name*, *lecture*, *title* with the roles „*student has name*", „*lecture has title*" and „*student studies (lecture*)".
In the context of natural language and information systems actually discussed are the methods, KISS [Kris94], RADD-NLI [ABDT95, BDTh96], Color-X [BuRi95].

The KISS method starts with a grammatical analysis. During this step subjects, objects and actions are extracted. These are used in the subsequent steps to generate the conceptual KISS scheme. RADD-NLI extracts requirements from questions expressed in natural language. According to the user's answers (also expressed in natural language) the conceptual scheme is constructed. Color-X integrates static and dynamic aspects and a formal specification of the requirements. The model for the latter (CPL [DiRi91]) makes use of natural language constructs which are defined in the so-called functional grammar [Dik78].

Even some of the methodologies for object oriented analysis (OOA [CoYo91, RBPE91]) give hints to extract classes from textual requirements. The problem here is, that this step is not strongly supported neither by these methodologies nor by their tools. Another lack of these hints is, that the designer is focused on classes only. We think that attributes and dynamic aspects are also worth to be collected.

Our approach was mainly influenced by the DATA-ID approach in that we use glossaries for scheme representation and some similar modeling notions. From NIAM we adopted the 'object-role' view for static UoD aspects that concentrates on identifying objects ('things') and their interrelationships ('connections') instead of forcing the designer to differentiate between entities and values or associations and attributes at this early design phase. This does not mean, that we forbid the designer to make that decision. If he wants to decide between attributes and classes, he is invited to do so - but this is optional.

3. KCPM: The Klagenfurt Conceptual Predesign Model

3.1. Basic notions of KCPM

An important aim for the development of KCPM was to harmonize the developer's and the end-user's view of a given UoD, i.e., to provide an interface for their mutual understanding. As mentioned in the previous section, the approach was influenced by the DATA-ID model which we extended and specialized to provide at least the same semantic expressiveness as modern approaches to object oriented analysis do. The most important modeling notions of the static part of our approach are: ***thing type***, ***connection type***, ***perspective*** and ***constraint***. We shortly explain these notions and give examples which show how to represent their usage within glossaries.

Thing type
Generalization of the conceptual notions *class* (entity type or entity set, respectively) and *attribute*. Thus, typical things (instances of thing types) are

- natural or juridical persons,
- material or immaterial objects,
- abstract notions.

as well as

- descriptive characteristics of the above mentioned examples (e.g. a customer name, a product number, a product description).

In contrast to that DATA-ID distinguishes between entity types and attributes but we did not want to force a requirements engineer to decide on that at such an early stage of the analysis.

Apart from the simplification this generalization induces for the end-user (in whose world everything just continues to be a „thing"), it also provides for more flexibility during linguistic analysis. For, although there are heuristics and rules to find attributes in natural language sentences, these do not work in all situations. In undetermined cases, however, using KCPM we can defer the decision to later transformation steps. Moreover, the known conceptual models have different conceptions concerning which relations between attributes and classes, between two or more classes, and between attributes they permit. Hence design decisions often depend on the particular modeling concept of the model in use. The generalized KCPM approach allows to separate these model specific aspects from the linguistic analysis because there is no a priori restriction.

Things are related within the real world. To capture this, we introduce the notion of ***connection type***. Two or more thing types can be involved in a connection type. To define a connection type completely, it must be described from the point of view (***perspective***) of all of the involved thing types. This corresponds to the NIAM object/role model.

Sentences leading to connections (and perspectives) are e.g. the following:

(S1) *a flight has passengers*. (perspective of *flight*)
(S2) *a passenger books a flight*. (perspective of *passenger*)

The abstractions *generalization* („is-a", with set inclusion on the instance level) and *aggregation* („is part of") are treated as specific connection types.

Static UoD aspects that cannot be modeled using these notions (and their characteristics, see below) are captured by (textual) constraints. This is not very sophisticated, but allows the designer to specify functional requirements as well as non functional requirements [Somm89]. The glossaries form the basis of an information resource dictionary for the application in question and emphasize the scratch pad character. Traceability between all the entries and the requirements source is given by the means of so-called requirement source references. In the default case, these references refer to natural language sentences. Meta attributes are used for characterizing aspects of our basic notions. These are represented as glossary columns. In addition, each glossary entry is equipped with a unique identification number.

Relevant thing type meta attributes
name: Thing type designator, directly extracted from the requirements document.
classification: information about the conceptual „equivalent" (e.g. entity type, attribute) if extractable from the text. If it is not extractable it will be determined during the mapping process from the pre-design to the design scheme.
quantity: the expected number of instances of the thing type, e.g., derived from a sentence like

(S3) *we have **500** employees.*

examples: instance denominators of the resp. thing type (these are not treated as instances of an independent denominator type), derived from sentences like

(S4) *Mr. Miller, living in Klagenfurt, is a typical customer.*

Value domain: natural language description of the legal value domain, e.g. from sentences like

(S5) *the age must be over 30.*
(S6) *deliveries are possible all months except July.*

synonym: references synonyms (e.g. *passenger*, *customer* in a specific airport UoD).
description: narrative text on the resp. thing type.
requirement reference: a pointer to the (natural language) documents/sentences where the resp. thing type is referenced.

Relevant connection type meta attributes
name: may be derived from one of the perspective names (see below).
description: narrative text describing the connection type.
Connection type determiner: references, if it exists, an „objectification" of the connection by means of an existing thing type. Consider, e.g., the sentences

(S7) *The baggage is assigned to the corresponding flight.*
(S8) *The ground personnel checks the assignment.*

In (S7), the perspective *assigned* could be seen as a verbal description of the connection. In (S8) the word *assignment*

references a thing type which takes part in a connection with the thing type *ground personnel*. It seems that *assignment* also references the connection *assign*. In this case, the relation between *assignment* and *assign* should be made explicit. This is done using the connection type determiner, which is itself a reference to a thing type.
additional cardinality: cardinalities which cannot be expressed in one perspective (see below).

per perspective
perspective name: name of the perspective - normally the verb of the sentence in its active or passive form.
involved thing type: thing type connected via the responsible perspective.
optional: If there are situations where the involved thing type needs not to be instantiated as part of a connection, this value is set to *True*.
cardinality (min, max): relation qualifier similar to numerous conceptual models: Each cardinality has a lower and upper bound. The lower bound can range between zero and a definite number. The upper bound must range between the lower bound (must have at least the value „1“) and an indefinite value „N“.

UoD-area: airport		UoD: airport	department: --					
id#	name	classification	quantity description	examples	value domain	synonym	description	requirement source
D001	airport	thing-type						S9
D002	passenger	thing-type						S1, S2
D003	customer	thing-type		Mr. Miller		D002		S4
D004	flight	thing-type						S1, S2, S7
D005	captain	thing-type						
D006	employee	thing-type	500					S3,
D007	age	thing-type			over 30			S5, S9
D008	baggage	thing-type						S7
D009	ground-personnel	thing-type						S8, S9
D010	assignment	thing-type						S8
D011	delivery	thing-type			all months except July			S6

Figure 1 (part of the thing type glossary)

perspective determiner: circumscribes the involved thing type of this perspective, e.g. derived from sentences like

(S9) *Some employees work on airports as ground personnel.*

In this case *employee* and *airport* are the involved thing types. The thing type *ground personnel* is seen as a perspective determiner [FKM97]. The figures 1 and 2 show the corresponding glossary entries for the example sentences *(S1)* to *(S9)*.

A meta scheme specification of these notions is given in figure 3. The key elements of the model (thing type, connection type, perspective and constraint) are modeled as classes. Since they have been explained before together with their (meta) attributes we now concentrate on the association between them and on some further concepts. As has been pointed out, there may be references from a connection type to thing types, namely *perspective determiner* and *connection type determiner*. Both are represented by associations between thing type and perspective as well as thing type and connection type respectively. The *involved thing type* of a perspective is also a reference to a thing type. Hence an association with the same name exists. A perspective is characterized by its name and by its minimum and maximum cardinalities.

UoD-area: airport				UoD: airport	department: ---				
c-id#	name	connection type determiner	de-scription	perspective					require-ment source
				perspective#	involved thing-type	name	min, max	perspective determiner	
C001				p001a	D004 , flight	has			S1, S2
				p001b	D002 , passenger	books			
C002				p002a	D006, employee	work		D009, groundpersonnel	S9
				p002b	D001, airport				
C003		D010, assignment		p003a	D008, baggage	is assigned			S7
				p003b	D004, flight				
C004				p004a	D010, assignment				S8
				p004b	D009, ground-personnel	checks			
...									

min, max: minimum, maximum cardinality.

Figure 2 (part of the connection type glossary)

If the case of n-ary connection types there might be ***additional cardinality*** constraints between subsets of the involved thing types. Think e.g. of a connection type *delivery* that connects the thing types *customer*, *product* and *vehicle*. A functional dependency *customer* x *product* -> *vehicle* cannot be expressed by perspective cardinalities but by a connection type specific ‘additional cardinality’. *Customer* and *product* are the ***sources***, vehicle is the ***target*** of that cardinality.

3.2. Design environment

To support traceability between requirements and *design objects* (thing types, connection types and constraints) these are related to their ‘*requirement sources*’ (see figure 4). We distinguish between *textual* and *other documents*. Textual documents consist of *parts*, document parts are supposed to consist of *sentences*. The distinction mainly has been made for supporting linguistic analysis, a further specialization of the other documents (graphical, voice etc.) could be introduced if necessary.

Designer questions and *remarks* is another feature of our scratch pad oriented approach. While collecting and analyzing requirements the designer might use that feature to note open questions or remarks.

Each design object is part of an *organizational environment* (e.g. of a department) that again is part of a certain *universe of discourse*. Each universe of discourse can belong to one or more so-called *discourse areas*. Several UoD's belonging to the same discourse area share some common aspects. E.g., two or more UoD's could belong to the same branch, or, some tasks within the UoD's are equivalent [BDTh96] like borrowing books/media and car rental, or, because of other reasons.

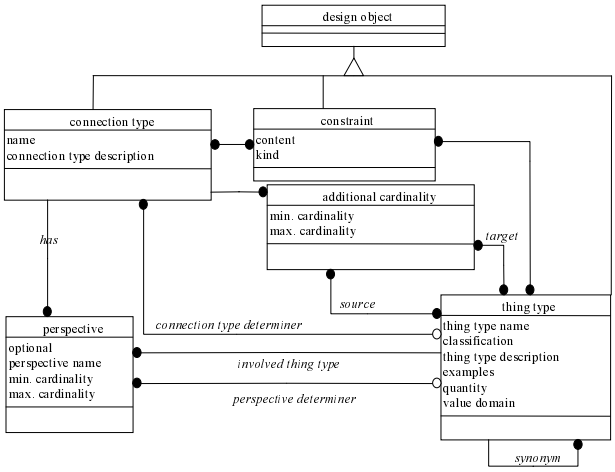

Figure 3 (KCPM meta scheme)

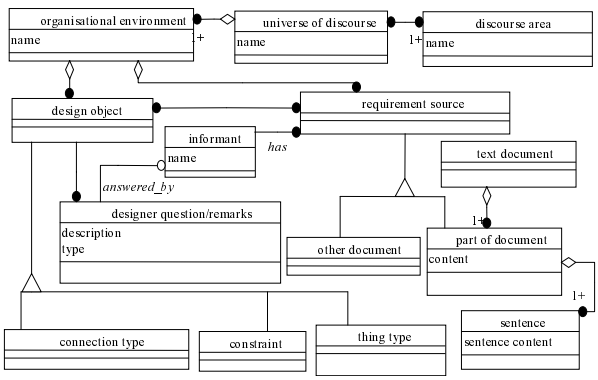

Figure 4 (scheme of the design environment)[3]

[3] The notation again corresponds to that of OMT. Examples for how to read the scheme are the following:
A universe of discourse consists of one ore more organizational environments (part of!)
A connection type, a constraint, a thing type IS A design object
A design object belongs to 0 or more requirements sources and vice versa.
A requirement source has exactly one informant/An informant is responsible for 0 or more req. sources.
A designer question can be answered by 0 or 1 informant.

4. Applicability and tool support

4.1. Practical Experiences

Corresponding to our background in business application domains, KCPM is intended for the 'pre-design' of business information systems (BIS). It seems to be very likely, that it might be useful for requirements specification within other application domains, too. This, however, is still to be investigated.

First extensive experiences in using KCPM were gained from a project run by an Austrian newspaper producer: The project's goal was to identify the requirements for a new business information system to support the complete sales organization including the control of newsmen, the newspaper distribution etc. In order to get the requirements as completely as possible, a participative approach was been chosen integrating employees concerned of all levels and departments: from the head of the sales department to the stock clerk, from administrative people to the newsmen.

It turned out that all of them, though using the KCPM notions rather intuitively, did this in a correct way. They did so without having experience in KCPM nor having a specific initial training. In accordance with the enterprise only short introductions into the glossary structure were given at the beginning of the interviews.

Especially the description of static UoD aspects posed no problems at all. Against that, for gaining requirements concerning the application dynamics, the initial KCPM glossaries had to be simplified to some extent.

Several hundreds of thing and connection types were identified within the requirements collection phase. Their validation and consolidation (i.e., the integration of the views of different persons) could be done rather easily. Problems arose only from the fact that at that moment only a simple MS-Excel based tool was available for handling glossaries. This, however, is not sufficient when dealing with large numbers of thing types etc. (think, e.g., of mechanisms for homonym and synonym detection, graphical representation, reorganization following clustering decisions etc.).

The glossaries resulting from that project were used as the basis for contracting the system realization (with a well known German BIS supplier).

4.2. Scalability

At a first glance, our approach seems to let explode free text into much larger tables, thus arising the question of scalability. Clearly, tables are physically more spacious than free text. On the other hand, they allow for a structured representation of complex and voluminous information that may be organized, manipulated and analyzed using computerized methods.

(If we disregard, e.g., the semantic modeling aspect, a KCPM scheme might be seen as a special kind of data dictionary with all its advantages.) On the other hand, a closer look at the tables of figure 1 and 2 shows that there are a lot of empty table fields reflecting that necessary information is ***not*** referred to in sentences (1) to (9). Thus, further sentences are necessary (i.e. asked by a system analyst recognizing the empty fields), that would not lead to further glossary lines but to a more complete description of the requirements.

4.3. The NIBA project

Actually we are re-implementing our first prototype supporting the handling of KCPM glossaries. In addition to that we run the joint project NIBA with the department of computer linguistic of our University. The goal of NIBA is to automate, as far as possible, the process of producing pre-design schemes by extracting their entries from the end-user's natural language requirements statements. Our approach uses as a linguistic basis the natural language grammar model NTS (Natürlichkeitstheoretische Syntax) [MFW98]. This grammar model is based on generative syntax, and is extended to semantic characteristics. A parser is available, which transforms a sentence into a tree structure. Using this tree structure we derive from natural language phrases entries of the predesign scheme. Work has been done so far to derive the glossary concepts thing type, perspective and connection type, cardinality, perspective determiner, connection type descriptor (see [FKM96, FKM97]).

Another aspect is the mapping from conceptual predesign to conceptual design. This is necessary to get a formalized view of the UoD. We do this in several steps. First a check is made, whether the glossaries entries are consistent, then the glossary entries are mapped to the entries of the conceptual schema. In our prototype system this second step (mapping of the glossary entries) was implemented, to show the correspondence between the glossary entries and conceptual schema entries.

5. The mapping process

For the derivation of a first cut OMT-Scheme from requirements documents we propose an iterative process model consisting of two phases (*predesign* and *transformation*) with three steps each.

- **predesign steps:**
  - glossary initialization
  - entry collection (=modeling using KCPM notions)
  - consistency and completeness check
- **transformation steps:**
  - mapping to conceptual notions
  - restructuring
  - completion

We will discuss these steps in some detail in the following subsections.

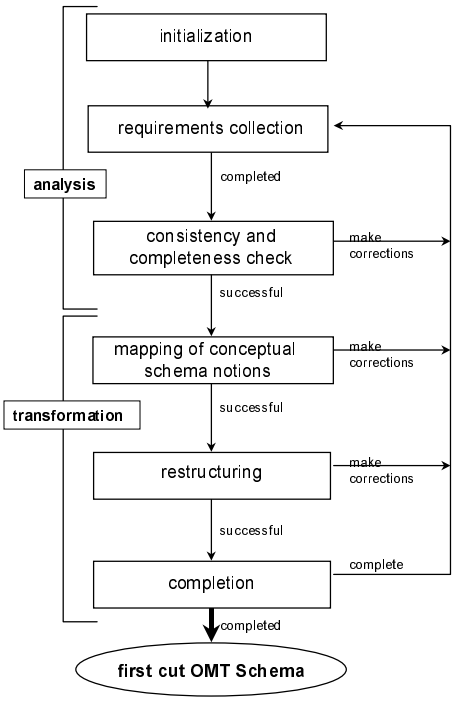

Figure 5 (mapping process)

5.1. Initialization and information collection

The first step, *glossary initialization*, relies on the assumption, that a designer knows something about the organizational environment of the UoD. He may want to fill in these known information before he interviews end-users. Of course, the first phase is optional or might be done automatically (reuse out of a design knowledge base, e.g. consisting of generalized discourse area specific glossaries). For a more detailed description see [Mayr95].

In the *information collection* step the designer adapts existing glossary entries and collects new ones from requirements documents (generated e.g. via interviews) until all involved requirements sources are 'exhausted'. The mapping part of this step is, as has been mentioned before, addressed in our NIBA project.

5.2. Consistency check

This step can be formalized to a large extent and thus again be deferred to a computerized tool: the contents of the glossaries are to be checked whether they are consis-

tent and structurally complete. Any detected inconsistency or incompleteness must be corrected by the designer.
The completeness check addresses mandatory values for meta attributes, connectiveness (no isolated thing types), perspective cardinalities, names etc..

An example for inconsistency checks is the search for cycles established by aggregations or generalizations.
Another check addresses the identification of synonymous thing types or connection types. If, e.g., two or more connection types share the same involved thing types, then it should be detected and presented to the designer. This concept is known from conceptual schema validation, see [Stor91].

5.3. Mapping to conceptual notions

This step again may be automated to a large extent on the basis of a predefined set of mapping rules. Clearly these rules depend on the model chosen for conceptual design (OMT in our case). We distinguish ***laws*** and ***proposals***:

A ***law*** forces a specific mapping (e.g. a given thing type to a class). If the designer does not follow the law, the conceptual scheme will become inconsistent. A *proposal* means, that the proposed mapping is more likely than another one. Thus, the designer may accept the proposal or take another decision.

Furthermore, rules may divided into ***direct*** and ***indirect*** ones. *Direct rules* determine a target notion immediately from glossary entries. *Indirect rules* use, in addition, results of previous mappings.

Examples (target model OMT)

1. *A thing type maps to a class if it is involved in a connection type to itself (e.g. an employee manages other employees)*
This rule is a direct law, because OMT only allows classes to enter recursive relationships (associations).

2. *A thing type with a suffix like „*name“, „*address“ etc. in its name may map to an attribute.*
This rule is a proposal, because mapping to a class would not result into an inconsistent conceptual scheme.

3. *A binary connection type (two involved thing types) where one of the involved thing types previously was mapped to an attribute maps to a class.*
This rule is a law, since OMT does not allow for nested attributes. It is indirect since it refers to previous mapping results.
Based on the different kinds of rules, meta mapping rules as given in table 1 may be formulated.

Table 1 (meta mapping rules for mappings to the conceptual notions class and attribute)

	Class	attribute	mapping
1	law/proposal	---	class
2	---	law/proposal	attribute
3	Law	proposal	class
4	Proposal	law	attribute
5	Proposal	proposal	design decision
6	Law	law	contradiction

Table 1 has to be read as follows: If only rules for one alternative exist in a given situation, then the corresponding mapping has to be chosen (see rows 1 and 2). Laws overrule proposals (see rows 3 and 4). If only proposals are applicable in a given situation, then the designer has to decide on the target notion (see row 5). If conflicting laws are applicable, then the designer has to check for and to solve the reason for that inconsistency (see row 6). Figure 6 details the mapping step.

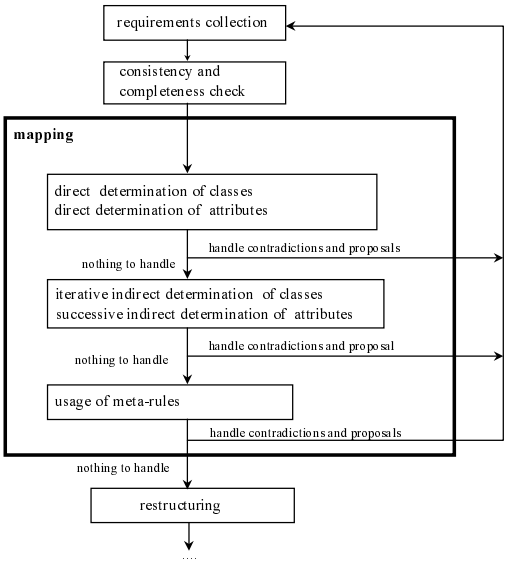

Figure 6 (mapping step - detail)

5.4. Restructuring and completion

Restructuring bases on consistency checks at the conceptual level. Typical checks are

- the check for orphan classes [Tauz89], i.e., classes which are not related to other classes,
- the check for classes with identical attributes. If such classes exist one has to decide if the classes are synonymous or if they establish a generalization hierarchy.

As a consequence of such checks changes on the predesign level, i.e. within the glossaries, might become necessary ('restructuring'). Typical restructuring candidates are the perspective determiners (see [FKM97] for more details).

Restructuring is followed by the completion step where incompleteness is detected by analyzing the conceptual scheme.

6. Conclusion

Conceptual predesign as proposed within this paper may bridge the gap between natural language requirements specifications and conceptual design. Using a lean but semantically powerful model like KCPM together with a glossary-based representation paradigm provides for an interface between end-users and designers that is transparent to both. This has been experienced in a rather large IS design project for an Austrian newspaper producer, where end-users with all kinds of educational backgrounds were involved.

Trivially, predesign schemes also might be visualized by graphical representations where necessary. More important is the fact, that models like KPCM seem to be an ideal 'interlingua' on the way from natural language requirements specifications to conceptual schemes. This is exploited in NIBA project, where automatic means are developed for deriving glossary entries from textual documents and for mapping these entries to conceptual schemes.

This paper concentrated on (notions and methods) for static UoD aspects. However, dynamic aspects are covered by KCPM as well. We actually work on the mapping of KCPM events to conceptual state diagrams.

Experiences with KCPM up to now exist for the application domain of business information systems. It is still to show, to what extent KCPM could also be useful within other application areas.

We finally want to express our thanks to the ICRE reviewers for their valuable comments that led to a substantial improvement of this paper.

References

[ABDT95] Albrecht, M.; Altus, M.; Buchholz, E.; Düsterhöft, A.; Thalheim, B.: "The Rapid Application and Database Development Workbench - A Comfortable Database Design Tool". In: *Proc. Int. Conference on Advanced System Engineering (CAiSE)*, Finland, 1995.

[BDTh96] Buchholz, E.; Düsterhöft, A.; Thalheim, B.; "Capturing Information on Behavior with the RADD-NLI: A Linguistic and Knowledge Based Approach". In: [RiBu96] pp. 185 - 196.

[BuRi94] Burg, J.F.M.; Riet van de, R.P.: "Color-X Event Model: Integrated Specification of the Dynamics of Individual Objects". In: [Papa95] pp 146-157.

[BuRi96] Burg, J.F.M: Riet van de, R.P.: Analyzing Informal Requirements Specifications: A first step towards conceptual modeling. In [RiBu96]. pp 15-27.

[Ceri83] Ceri, S. (ed.): *Methodology and Tools for Database Design*, North Holland 1983.

[Chen83] Chen, P.: "English Sentence Structure and Entity Relationship Diagrams". In: *Int. Journal of Information Sciences*, Vol. 29, 1983, pp 127-149.

[Chen91] Chen, P. (ed.): *Data & Knowledge Engineering*. North Holland Publ. Comp., Vol. 7, 1991.

[CoYo91] Coad, P.; Yourdon, E.: *Object-oriented Analysis*. Prentice Hall 1991.

[Dik78] Dik, S.C.: *Functional Grammar*. North Holland Publ. Comp., 1978.

[DiRi91] Dignum, F.: Riet van de, R.P.: "Knowledge base modeling based on linguistic and founded in logic". In: [Chen91] pp 1-34.

[Eick84] Eick, C. F.: "From natural language to good database definitions - a database design methodology". In: *Proceedings of the International Conference on Data Engineering*. Apr. 1984, pp 324 - 331.

[Fetr97] McFetridge, P. (ed.): *3rd International Workshop on Application of Natural Language to Information Systems*.

[FKM96] Fliedl, G.; Kop, Ch.; Mayerthaler, W.; Mayr, H.C.; Winkler, Ch: "NTS-based derivation of KCPM cardinalities: From natural language to conceptual pre-design". In: [RiBu96] pp. 222 - 233.

[FKM97] Fliedl, G.; Kop, Ch.; Mayerthaler, W.; Mayr, H.C.; Winkler, Ch: "NTS-Based derivation of KCPM Perspective determiners". In: [Fetr97]

[Kris94] Kristen, G.: *Object Orientation: the KISS Method - From Information Architecture to Information Systems*. Addison Wesley, 1994.

[KWDB86] Kersten, M. L.; Weigand H.; Dignum F.; Boom, J.: "A conceptual modeling expert system". In: [Spac86] pp 35-48.

[Loch89] Lochovsky, F. H. (ed.): *Proceedings of the 8th Int. Conference on Entity Relationship Approach*. North Holland Publ. Comp. Oct. 1989.

[Mayr95] Mayr, H.C.: "Conceptual Pre-design: A Platform for ReUse of Requirement specifications". Larry Latour, Kevin Wetzel (eds). *Proc.: WISR'95 - Intern. Workshop on Software Reuse*, St. Charles Illinois, August 1995.

[MFW98] Mayerthaler, W.; Fliedl, G; Winkler, Ch.: *Lexikon der Natürlichkeitstheoretischen Syntax und Morphosyntax*. Stauffenburg Verlag Tübingen, 1998.

[NiHa89] Nijssen, G.; Halpin, T. A.: *Conceptual Scheme and Relational Database Design - A fact oriented approach*. Prentice Hall Publ. Comp. 1989.

[Papa95] Papazoglou, M.P. (ed.): *The Proceedings of the Fourteenth International Object-Oriented and Entity Relationship Conference (OO-ER'95)*, Gold Coast Australia, 1995. Springer Verlag.

[PDF87] Pirow, P. C.; Duffy N. M.; Ford J. C.: "Information Systems in Practice and Theory". In: *Proceedings of the IFIP TC8 International Symposium on Information Systems*, North Holland Publ. Comp. 1987.

[RBPE91] Rumbaugh, J.; Blaha, M.; Premelani, W.; Eddy, F.; Lorensen, W.: *Object oriented modeling and design*. Englewood Cliffs, NJ, Prentice Hall, 1991.

[RiBu96] Riet van de, R.P; Burg, J.F.M; Vos, van der A.J.: *Application of Natural Language to Information Systems*. IOS Press Amsterdam, Oxford, Tokyo, Washington DC. 1996.

[Roll86] Rolland, C.: "An information system methodology supported by an expert design tool". In: [PDF87] pp. 189-201.

[Spac86] Spaccapietra, S.: *Proceeding of the 5th International Conference on Entity Relationship Approach*. North Holland Publ. Comp., Nov. 1986.

[Somm89] Sommerville, I.: *Software Engineering*. Addison Wesley Publ. Comp., 3rd edition, 1989.

[Stor91] Storey V. C.: "Relational database design based on the entity relationship model". In: *Data & Knowledge Engineering*, Vol. 7, 1991, pp 47-83.

[Tauz89] Tauzovich, B.: "An Expert System for Conceptual Data Modeling". In: [Loch89] pp. 205-220.