Visualizing Conceptual Schemas with their Sources and Progress

Christian Kop
Applied Informatics
Alpen-Adria-Universitaet Klagenfurt
Klagenfurt, Austria
chris@ifit.uni-klu.ac.at

Abstract - **Conceptual modeling for database design is more than just a "drawing" of the database architecture which is readable for specialists. Instead it must be a means for communication between the database designers and the other stakeholders. Even the specialists are not only interested in the graphical representation. There is also a need that the database designers and end users get an overview if the focus of the database schema under development still reflects the expectations of the end users. Stakeholders are also interested in the current working state (progress) of the model. Therefore, it needs simple and easy to use techniques for gathering and presenting different kind of information. In this paper, a combination of such techniques is presented. Firstly, it will be proposed how a glossary based representation together with a graphical representation and a verbalization of concepts can be used for communication with the end user. In the remaining parts of this paper these techniques will be applied to give database designers an overview of the focus of the schema, the current progress state as well as an overview to the sources which are related to the model elements.**

Keywords - ***conceptual modeling; verbalization; glossary; progress information; important concept;***

I. INTRODUCTION

A database is the backbone of information systems. Therefore, conceptual database design is a very important aspect of information systems development. Wrong conceptual models can lead to serious problems since the software depends on the right concepts and correct relationships between these concepts. Later changes in the database design can lead to numerous changes in the information systems software or to unforeseen errors. Much effort must be spent on the communication and negotiation process with all the stakeholders to get a validated conceptual database schema. Thus, it would be good to work with a presentation technique that is easy to understand and as many stakeholders as possible feel comfortable when using such a technique. Unfortunately this is not possible because of the different skills and knowledge of the stakeholders. Some of them are domain experts with no knowledge in computer sciences, others have a little knowledge. The problem is even worse since it is also situation depended. Thus, a single representation technique that is perfect for all stakeholders does not exist. A solution could be a mixture of representation techniques. Hence, the success of database projects strongly depends on a good mixture to gather the information from the end users as well as to present this information to them.

The most commonly used representation of conceptual models is a graphical representation. Since the beginnings of conceptual modeling (i.e. entity relationship modeling) models were represented with a graphical language (e.g., entity types as rectangles). This has not changed over the time. Some parts of the Unified Modeling Language (UML) have still a graphical language (i.e. classes appear as rectangles, associations as lines etc.). However, over the time computer scientists got aware that such graphical languages are good for IT professionals but typical end users are not able to understand them. Therefore solutions to verbalize the conceptual schema were introduced. Verbalization means that the graphical language is transformed back into natural language descriptions. Beside the classical graphical representation and verbalization, in this paper it is proposed that in addition a glossary representation should be considered as a third possibility. All these three representation techniques together can help the stakeholders to understand the conceptual schema.

For computer scientists, there are still good reasons to use graphical modeling languages. They provide a good spatial overview over all the concepts and their relationships. Furthermore a graphical language with a well defined grammar and defined notions is better suited to generate a logical model for the database.

Natural language descriptions of a diagram can better explain concepts and their relationships. Finally, if glossaries are used as check lists, they can support the negotiation process. Using these three representation techniques together can compensate the weaknesses of a single representation technique. Hence, the best solution would be to have all the three representation techniques under one roof. This can give all the stakeholders the opportunity to read that representation which is the best for them in a certain situation.

Most of the tools for conceptual modeling are focused on the graphical view. Some tools and approaches only provide at most two main views (graphical view and natural lan-

guage descriptions). Glossaries, natural language descriptions and diagrams together are not used in the context of database design, since most researchers rely on diagrams only.

Independent from the representation technique, the schema itself is only part of a greater design context. Every element within the schema must be traced back to a requirements source. During the design process, different elements in a schema will have a different working state. Whereas some elements are nearly completely modeled some elements still have to be finished. For some elements the designer must still ask questions or has open tasks in the task list, for other elements there are no more questions or tasks to do. There should be also the possibility to view concepts according to their importance in the schema. This is another kind of structuring mechanism to avoid that the stakeholders get lost within the network of concepts.

Therefore the paper is structured as follows. In Section 2, the related work is discussed. Section 3 gives an overview of two projects which were accomplished. Learning's of this project and the approval of previous research ideas and assumptions for the selection of the visualization strategies are presented in Section 4 namely graphical representation, verbalization and a glossary representation. Section 5 and 6 present additional visualization techniques based on the three basic visualization strategies. Section 7 shows parts of the tool. Section 8 summarizes this contribution.

II. RELATED WORK

Graphical representation (e.g., diagrams) is the most established type of representation for conceptual modeling in general and database modeling in particular. In the beginning of conceptual modeling, graphical languages like the Entity Relationship approach were proposed for both end users and database designers.

According to the underlying paradigm of how a stakeholder perceives the "world", two types of conceptual modeling approaches can be distinguished:

- Entity type and object oriented approaches,
- fact oriented approaches.

In the first paradigm the "world" is seen as a world of objects which have properties. Therefore a clear distinction is made between object and object types respectively and their properties. Representatives of this paradigm are the classical ER approach and UML. Fact oriented approaches on the other hand see the "world" as a world of facts. Facts describe objects and their roles within a relationship. No distinction is made between objects and properties. Every concept is treated equally. Representatives of this kind of paradigm are NIAM [14] and its successor ORM [8],[9]. Both approaches have pros and cons. Object oriented approaches look very compact. In a typical object oriented class diagram attributes are embedded in the class representation. No additional connections between classes and attributes are necessary which would expand the diagram. On the other hand, many revisions must be made if such a diagram is used too early in the design phase. Due to information that is collected, classes might become attributes and attributes might become classes. According to [8][9] this is a reason why fact oriented approaches are better suited for conceptual modeling.

Nowadays there are doubts that currently used graphical representations will support the communication between end users [13]. Therefore, it is proposed that more effort must be spent to produce good "diagrams" for user communication. Some researchers even state [4] [10] that the graphical representation of a conceptual model should be transformed back to natural language. In particular, they argue that this transformation better helps the end users to understand the very compact and sometimes formal graphical notation. As a solution for the transformation result, they often provide a restrictive form of natural language called controlled language [6]. Hence, the purpose of such a transformation (verbalization) step is to comment and explain the more formal graphical representation of relationships and concepts.

The use of glossaries and dictionaries was proposed since the 70. The first work on "glossaries" was done by Parnas [15]. He used tabular representations for the representations of functions. In the 80s the DATA ID approach [2] used glossaries as a central concept in their methodology. Requirements were distributed to data, operation- and event glossaries. The glossaries were the basis for traditional conceptual schema generation (ER diagrams and Petri nets). The KCPM approach [12] continues and extends this representation idea. It combines this idea with the fact oriented paradigm.

A similar technique to glossaries namely forms and templates were introduced for the description of use cases [3]. Another approach using form templates for functionality and navigation is NDT. It is described in [5]. In addition, the need for glossaries to describe also ontologies is proposed in [11].

Diagrams, verbalization strategies as well as glossaries can help to communicate with the stakeholders. Since the type of representation strongly depends on the skills of the stakeholder and the situation, a combination of all three representation techniques is always better as one representation alone. A lean modeling language which only consists of concepts and not of classes and attributes prevents that the database schema must undergo many changes.

Beside the communication to the end users it is also necessary that the designer knows the current working state within the model. Furthermore, he must know if each concept in the schema is related to at least one requirements source. Finally it would be good if he is supported in the question: "Do I still focus on the right things?"

Measures for the progress of requirements are given in [21]. These measures are based on the IEEE quality standards for requirements. Also in [20] an approach for meas-

uring the progress of requirements was discussed. This approach mainly depends on the decomposition of requirements and the number of statements like "*to be completed*". However, this could lead to two problems:

- when to end with the decomposition and
- forgotten "*to be completed*" statements

As a consequence, in [22] measurement is based on templates and not on natural language requirements as described in the two other approaches. Particularly glossaries are used. With this strategy the "to be completed" statements become superfluous. Hence, there is no problem if the designer forgets them. Instead any gap (empty cell) in the glossary is a hint for missing information.

The best practice to visualize the relationships to requirement sources is a traceability matrix [24].

Related research results which can help to determine if the designer still focus on the right things were found in the area of schema clustering [16][17]. In this field so called centered entities are used as a basis for the clustering. Other ideas were presented in the domain of ontologies [18][19]. Key concepts were mainly used to give one measure for the quality of an ontology.

To summarize the related work: Different representation techniques are proposed in literature. However, usually only one technique or a combination of two techniques is proposed. This paper proposes to combine the three representation techniques, namely a glossary based representation with a graphical representation and a verbalization. Furthermore it proposes to use the combination of these three representation techniques not only for the schema itself but for a specific content aspect (i.e. important concept) as well as for context information (progress information, relationship to sources). Hence, it is the aim that the stakeholders get a holistic view on the database schema.

III. PRACTICAL EXPERIENCES

Before the approach of different visualization techniques is described, two real projects are presented in this section as an additional motivation to the literature study. The two projects were accomplished in two different domains. The first project dealt with the management of cancer cases. Each province has an appointment from the government, that a central institution should collect the appearance of cancer cases. These are used by the government for statistical analysis. Usually a central institution located in one of the public hospitals takes care of this. The order was to support this institution during requirements elicitation and analysis. The institution worked already with an information system for managing cancer cases since the nineties. However, as the reader can imagine, within ten years, knowledge about cancer cases has grown and requirements of managing data and especially statistical data about cancer has changed. Therefore it was necessary to develop a new system.

The second project is located in the area of electrical power plants (mainly hydroelectric power production). A central institution monitors all the plants in the province. It checks if all plants work correctly and it has to react if an accident happens (i.e. to assign a team to fix the problem) or the plant is switched off (e.g., because of maintenance). The crew which monitors all the plants has to note all the events so that there is a traceable logged documentation if there is a shift changeover of the crew. Also for the management it is interesting to see what is going on, which accidents happened and the reasons for switch offs. Although the monitoring crews have access to different data sources, they need a summary of all these information in a central database.

Beside their differences, both projects can be characterized by the following similarities:

- The projects had a strong data centric aspect. Data was needed to get statistical information and to support the decision making in both cases. Conceptual modeling to design the new database and communication with the stakeholders were important tasks.
- The project was not built from scratch. Either the data in the old system (cancer cases project) had to be considered or the new system has to gather and "summarize" data from different data sources (power plant project). However in both cases there was not such an amount of data that the development of a data warehouse was justifiable.
- In particular, it was also necessary and useful to analyze the type of data available in the old database or other data sources.
- In both projects the stakeholders agreed that a new system with new features is necessary. For the "cancer case" project, the old database system was outdated. Only those data which has proved to be interesting over all the years was kept together with new information that was needed because of the new knowledge. For the power plant systems the stakeholders needed a new database system which stores the integrated data from the different data sources.
- Because of the different skills, background and knowledge of the end user it was not possible to describe the needed data with class diagrams only.

Especially the last mentioned similarity underlines the proposals found in literature and was a motivation to think about a combination of three representation techniques and to apply these techniques also for specific purposes (i.e. progress information, relationship to sources). Since the two projects were data centric, the remainder of the paper focuses on visualization strategies for a conceptual database schema and will not discuss any other aspect of a software system (i.e. function, behavior, user interface, non functional requirements etc.)

IV. THE THREE VIEWS

A. The model elements and graphical representation

Before describing the several views the model is briefly introduced here. It is based on the ORM paradigm (facts instead of entity types). Therefore no distinction between classes and attributes is made.

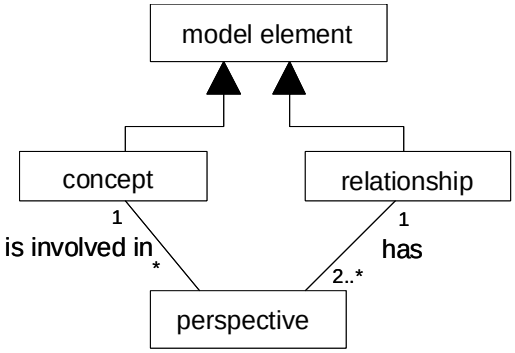

Figure 1: excerpt of the meta-model

The excerpt of the meta-model in Figure 1 illustrates this fact oriented paradigm. A concept is connected to a relationship via perspectives (roles in ORM). Both concept and relationship are model elements.

A concept itself is every term which is important in a certain domain. A concept can be a material or immaterial thing. It is also a term which would be modeled as an attribute in UML (e.g., first name). This supports the idea that designers shall elicit important concepts without thinking if they will become classes or attributes. Such distinction can be delegated to the tool.

Although the meta-model follows the fact oriented paradigm which allows that a relationship has more than 2 perspectives (e.g., ternary relationship) the representation of relationships is more similar to UML. Perspectives (roles) are hidden in the representation of a relationship. They are mainly used to specify the relationship. In this aspect it differs from ORM which strongly focus on roles also in the graphical representation. Especially special relationships (e.g., aggregation) are defined by pre-defined perspectives (e.g., aggregate_of, part of). Beside the well known relationships like aggregation, composition and generalization also an identification relationship and a hasProperty-relationship are part of the approach. The hasProperty-relationship which was introduced in [23] can be used to indicate that A has the property B. That B is a property of A does not necessarily mean that B is an attribute of A, if A and B would be mapped to an UML class diagram. B will only become an attribute if B does not have relationships to any other concept in the schema. On the other hand A can be transformed immediately to an UML class since it was specified with the hasProperty-relationship that A has a property. With this relationship alone a graph of concepts can already be easily transformed to an UML class diagram.

The hasProperty-relationship is drawn with a directed edge pointing from the object representative to the representative of the property. Whereas the perspectives are predefined ("*has*", "*belongs_to*") the whole relationship can be labelled individually. The "*identifies*" relationship is used if the designer knows that the value of a concept identifies another concept. The predefined perspectives of this relationship are "*identifies*" and "*is-identified-by*". The whole relationship is presented as an edge with two lines crossing the edge at the position of the identifier. With the two crossing lines, the relationship should appear like a "*key*". The crossing lines represent the teeth of the key. The identify relationship must be used to model concepts (attributes) which will become key candidates in the database schema. If no special relationship is applicable, then also a (simple) binary relationship can be used with no special meaning. It is represented as an edge with no additional graphical features. The user freely can label the two perspectives as well as the whole relationship or leave the labels empty. Figure 2 shows the appearances of the different relationships.

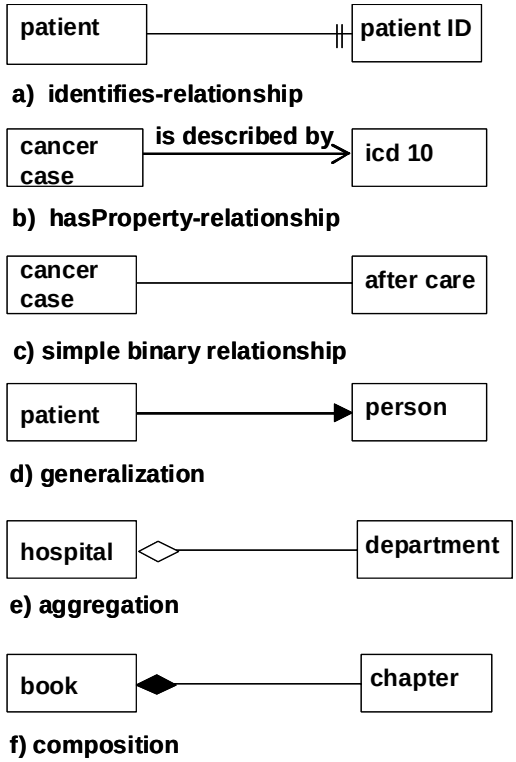

Figure 2: representation of relationships

Multiplicities must be defined for the normal binary relationship, the aggregation and for the *hasProperty*-relationship. There is no need to specify the multiplicity at the composite perspective of a composite relationship since the composite relationship has the same multiplicity seman-

tic as the composite association in UML. There is also no need to specify multiplicities at all for the "*identifie*s-" and "*generalization-*"relationship. Because of their special semantics the multiplicities are implicitly defined.

Another difference to UML, ER diagrams and ORM is the management of additional concept information (e.g., examples, quantity descriptions, synonyms, value constraint). Since this information is well suited for a glossary view, it will be described in detail in the section which treats the glossary view.

Because of the semantic relationships, the information gathered in the glossaries (e.g., value constraint), information about multiplicities, concept name analysis and relationship name analysis, the approach allows an easy transformation to an Entity relationship or UML diagram. Hence, like ORM the approach is stable against changes in the model but can be transformed to UML. For a more detailed description of the model and the mapping, the reader is referred to [12] and [23].

B. Glossaries

Glossaries should not compete with the other representation techniques but try to complement them. Whereas graphical representations are good for a (spatial) overview and natural language descriptions explain formal notations to end users, the aim of a glossary should be a detailed but compact description of concepts. They should provide the negotiation process and also the process of collecting concepts from the stakeholders. Especially during the first stage of database development, a database designer is more like a medical doctor or a pilot who must work with check lists in order to get new information or validate old information. With glossaries the collected concepts will appear in a very compact format which is still readable and understandable for all the stakeholders.

Most often a concept glossary only has a column for the name of the concept and a column for the definition of that concept. With this kind of information a glossary would only play a minor role.

With additional glossary columns, different stakeholders can be incorporated (e.g., typical end users with no technical knowledge and persons with technical knowledge about the old system).

For instance, in the cancer case project there was a person who maintained the old system and of course the typical end users like physicians, nurses and secretaries. In the electrical power project, a project member from the customer's side had knowledge about the existing data sources from which the data should be extracted. If it is interesting in particular to represent information for all the stakeholders like it was in these projects, then such a glossary must not only consist of a concept name and definition column. Instead the following additional columns are necessary:

- Examples for the concepts,
- quantity description,
- synonyms,
- value constraint,
- data source constraint.

Figure 3 shows the part of the meta-model that manages a detailed description of concepts. It is visualized with the model elements presented here.

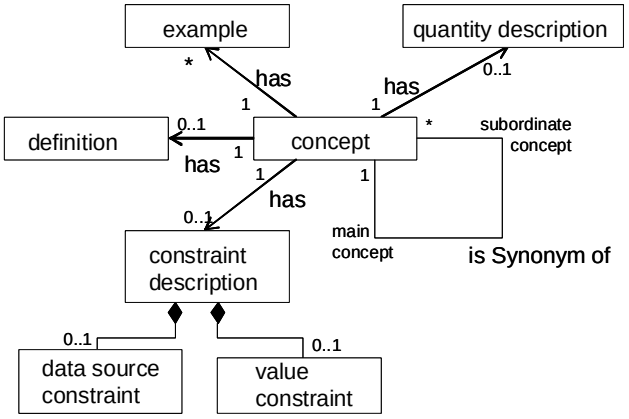

Figure 3: concept information

According to its name, instances and values are stored in the **example** column. (e.g., "*pathological institute*" for the concept "*department name*").

Quantity description specifies the amount of instances a concept will have (e.g., "*500 patients*"). It can be further refined with an indicator that tells if it is an average, a minimum or a maximum value (e.g., "*in average 500 patients*"). Additionally, with a second descriptor it can be specified that the quantity increases within a certain period (e.g., "*10 additional patients per year*").

Synonyms refer to other names of the same concept. (e.g., *institute* as a synonym for *department* if department was chosen as the main working concept). Usually synonyms have no internal hierarchy or ordering. If a notion N1 is synonym of a notion N2, then also N2 is synonym of N1. For conceptual modeling it is necessary to decide which concept will be further used. This is selected as the main concept in the list of synonymous concepts. The other concepts are still necessary but only in the sense that they represent variants of the main concept. Therefore in the synonym relationship of the meta-model the perspectives (roles) main concept and subordinate concept were introduced.

The **value constraint** consists of the sub information *format* and *data type*. The data type column specifies the data type a value can have. It can be a simple data type (String, Integer, Date etc.) or an Enumeration. It is intended that also smart business objects can appear in this column. Smart business objects as proposed in [7] are specific data types which are restricted to a certain format and specific features and operations (e.g., a type "*e-Mail address*"). In the format column, the appearance of the values is specified. The simplest form is the definition of the length of a value

(e.g., a String or an Integer value). The length is encoded with the character L. Thus, a string value with 50 characters is encoded with L50 and has the entry "String" in the data type column. If a concept like diagnosis date is based on the data type "DATE", then this data type appears in the data type column. If such a data has a specific format (e.g. "YYYY/MM/DD") then this is collected in the format column.

The **data source constraint** consists of the same information. In addition it has the column *data source*. The *data source* constraint column was introduced since often an old database system exists which has to be replaced by a new one. It cannot be expected, that the data types and formats will stay the same in the new version of the system. If data from the old system is migrated into the new system, possible differences of data types and formats must be considered in advance. The additional column "data source" specifies the source and the structure of a concept in the old system from which data has to be migrated. The expression "<table name>.<attribute>" is used for it. If the concept is only a table in the old database, then only "<table name>" can be used. Table 1 in the appendix shows how such a representation can look like.

C. Verbalization

For verbalization it is assumed, that class names and concept names respectively are in singular form. Relationship names are verbs in 3rd person singular form. These verbs can be either given in active or passive voice.

Usually a verbalization of a diagram is made by paraphrasing the graphical content. In particular, the approach described in [10] uses the label of object types, the labels of the roles and the multiplicity information. This information is concatenated together with fillers (e.g., articles, quantifiers) to form a natural language sentence. Especially the multiplicity information must be translated from a number representation (e.g., numbers in brackets [0..1]) to its textual representations. Short cuts like "*exactly 2*" for [2..2] must be considered.

As mentioned above a concept name is written in singular in the graphical representation. In the resulting sentence of a verbalization, it can be left in singular if it is the subject of the sentence. If it is the object in the sentence then it has to be decided if this concept name must be transformed to the plural form. The decision is based on the multiplicity information (e.g., N as the maximum multiplicity).

The verbalization strategy also takes the special relationships between the concepts into account. Beside the commonly used special relationships like "*Generalization*" and "*Aggregation*", the model also offers the special relationships "*hasProperty*" and "*Identification*". These additional special relationships make it easier to verbalize the graphical representation of the relationship. In the "*hasProperty*" relationship the verbs *has*/*belongs_to* are taken as default paraphrases for the relationships between the concepts, but the designer always can use another word (e.g., *owns*, *buys* etc.) instead of *has*. In this case the word with which the user defines the relationship is taken in the verbalization step. The special relationship "*Identification*" provides two roles. These roles are *thing identifies (another) thing* and *(another) thing is identified by thing*. They are taken for verbalization. The place holder "*thing*" is replaced by the concrete involved concepts of the identification-relationship. If no special relationships are used, then it is recommended, that the user specifies the name of the relationship. Otherwise, the relationship is verbalized into a simple "*is related to*" phrase.

In addition to relationship verbalization, also a verbalization of some of the concept columns (quantity description column, format column and value constraint column) is provided. Special sentence pattern are used. A sentence pattern like "*There are [in average | at least | at most] <quantity> [additional] <concepts> [per year]*" can already support the verbalization of a quantity description. The phrases in square brackets are optional. The minimal specification of a quantity is "*There are <quantity> <concepts>*". This is equivalent to "*There are in average <quantity> <concepts>*." If there is an upper limit that can be reached, then "*at most*" is taken. If the quantity will never fall under a minimal limit then "*at least*" is used. If not the total quantity is meant but a quantity that rises per year then "*additional*" together with "*per yea*r" is added.

To specify the format column a sentence like "*The format of the <concept> is <format description>*" can be used. If enumerations are defined in the value constraint column of a concept a sentence like "*<concept> must be either <value> or <value> or <value> …*" is generated.

V. VISUALIZING IMPORTANT CONCEPTS

In the last section the three basic presentation strategies were introduced. This section builds on the three presentation strategies. They are used to visualize important concepts. Information about important concepts is useful to get a quick overview of the schema focus. Especially the two questions are of interest:

- Is the focus of the schema still the focus which was expected by all the stakeholders at the beginning of a database design project?
- Is a certain concept specified enough?

These questions can be broken down to the question of important concepts within a schema. If the important concepts modeled in a schema are not the same as expected by the stakeholders, then it is possible to detect a defect. For instance, such a situation can appear if an important concept is still underspecified. This can happen due to a misunderstanding between the designer and end users. Particularly, the designer concentrates on the description of concepts which are not so important for the stakeholders.

In order to get this information, it is necessary that the tool itself can automatically determine important concepts on the basis of already modeled information. An adjustment can then be made between the generated proposal of the tool and the expectations of the stakeholders.

This section will discuss this topic. After defining what important concepts are and how they can be calculated, it will be explained how the different views can visualize this kind of information.

A. What are important concepts?

The notion "important concept" is based on the idea, that they are well described in a conceptual schema. They make up the centers of the schema and other concepts (supporting concepts) are used to describe them. Synonymous notions for important concept are "centered entity" and "key concept".

[16] has introduced the notion "centered entity" for using it as a basis for a clustering algorithm. Entities are described in terms of relationships in which they are involved. Hence, the more an entity has connections to other entities; the more the entity can be seen as a centered entity. This definition of a centered entity is very pragmatic, based on the analysis of a graph. It has the advantage that it can be done automatically by the tool [16], [17].

Most important for the approach introduced here is a research result achieved by the same author some years later [17]. A study with students was made. One part of the study focused on the centered entities itself. The question was examined, if entities with more relationships are perceived as more important. The study showed that this is the case.

In the research area of ontologies the notion "key concept" was introduced in [19]. It was part of an approach which checked the quality of an ontology. Once again relationships were used for the calculation of key concepts. Here, the relationships are weighted higher, if more implicit relationships in the lower sections of the generalization hierarchy can be derived from them. In [18] only the children of a concept in a specialization hierarchy were counted.

B. How to calculate importance?

Since database design is more focused on relationships between concepts than on a generalization/specialization taxonomy, this approach follows the idea of [16][17]. It differs and extends this previous approach since it considers the type of relationship between the concepts. The calculation consists of two sub steps:

- Counting of connections to other concepts
- Categorizing a concept.

Counting step: For the approach presented here, the counting is done as follows: For each binary undirected relationship a concept has, the counter is increased by 1. For each special generalization relationship a concept is involved, the counter of that concept is increased by 1. If a concept is involved in the special aggregation relationship then the counter is increased only if it has the aggregation role. This is based on the idea, that aggregates more represent the main concepts than their parts since otherwise it would not be necessary to model the aggregate but it can be concentrated on the parts only. The two additional semantic relationships *identification* and *hasProperty* are also counted differently. The counter is increased for a concept if it is identified by another concept. The counter is not increased for the concept which identifies, since this concept can be understood as a (database key candidate) attribute. The *hasProperty* relationship is a directed relationship between a concept and its property representation (once again a concept). For each *hasProperty* relationship where the concept is in the role to have the property and not in the role to be the property the counter is increased by 1.

Counting in other approaches (UML, ER, ORM): For the sake of completeness, the step is also explained for UML, ER and ORM schemas. The counting of importance depends on the paradigm which is used.

For UML or ER the counting could be as follows: All attributes in an UML class diagram or Entity Relationship diagram get the count value 1. For each class, entity type respectively, their numbers of attributes are counted. For instance, if a certain class (entity type) has 12 attributes, then its initial count result is 12. For each binary (n-ary) undirected association, a class / entity type is involved in; the count result is increased by 1 for that class / entity type. For each generalization relationship a class / entity type is involved in, the count result is increased by 1. For each aggregation- or composition relationship a class / entity type is involved in as an aggregate the count result for that class / entity type is increased by 1. UML class diagrams provide two additional features, which are interesting for counting. Associations can be extended with a reading and navigation direction. In these two cases the count result is increased by 1 only for those classes which are the source (starting point) for the reading or navigation direction. It can be argued that the source of the navigation or reading direction is focused. Hence, it is more likely that it is an important concept than the target of the directed association.

For an ORM diagram the counting is as follows: For each role of an object type, the counter is increased by one. If the object type in addition has a key attribute, then the counter is increased by 1 once more. Aggregation is treated in the same way as shown for UML.

Categorization step: The result of the counting for each concept is now taken as an input for the categorization step. Additionally the concept with the maximum counting result is selected out of the list of concepts. This is the first detected important concept. The counting results of all other concepts are compared with this maximum.

The comparison returns to which category a concept belongs. The approach is restricted to the three categories:

v*ery important concepts*, *important concepts* and *unimportant (supporting) concepts*.

The distinction into which category a concept falls is determined by the percentage of counted connections a concept has with respect to the maximum counting result in a certain schema. If a concept reaches a percentage value >= 66 % then it is a very important concept. If the percentage value PV is 33 % <= PV < 66 % then it is an important concept. Finally, if the percentage value is below 33 % then it is only a supporting (unimportant) concept.

Let a concept C_1 have a count of 50, meaning it was able to increase the counter by 1 for 50 relationships it is involved in. Let us further assume 50 is also the maximum counting result that appears in this schema S. Let another concept C_2 in S have a counting result of 20. The type of the concept is then calculated by 20 / 50 and the result is 0.4 (40 %). With this 40 % the concept belongs to the category of important concepts. Let a third concept C_3 have the value of 40 which means, it reaches the maximum with 80 %. Hence, C_1 and C_3 belong to the very important concepts.

After this introduction what main concepts are and how they can be detected in the schema, the next section discusses how such information can be offered to the user in the different representation techniques presented in this paper.

C. Visualization

For the **graphical representation** a strategy was chosen, which is a combination of enlarging the rectangular dimensions of a concept together with a coloring strategy. The very important concepts appear as the biggest concepts on the screen. The color of this concept is deep green which emphasize their importance. Important concepts are also enlarged but not as much as very important concepts. They appear in a yellow color. This gives them a more transparent and pale touch. The color and the size signalize that they must be considered as important, but they are not among the most important. Finally the supporting concepts are not enlarged at all, but appear as they are. They have a white color, which underlines their supporting character. The spatial information is not distorted as it is only necessary to show which concept is very important, important or unimportant.

In the **verbalization** view all unnecessary information is filtered out to avoid textual bulk. Like in a news paper, book chapter or any other linear textual description an abstract or summary of what has been specified is provided to the reader. For those kinds of concepts which are important according to their specifications the user gets a very detailed and insight look. On the other hand he will not be bothered with details of supporting concepts. They only appear in the textual summary as long as they help to describe at least one of the (very) important concepts. Such a verbalization can start with a textual introduction template like: "*The most important concepts of this schema are <list of very important concepts> followed by <list of important concepts>*". Afterwards each of the (very) important concepts is verbalized according to the strategies described in the verbalization section.

The **glossary content** can be sorted. For sorting, an additional glossary column is introduced. In this column, the counting results are presented. If the glossary rows are sorted according to these columns in a descending order, then the very important concepts appear before the important concepts and the supporting concepts.

To summarize, if for instance "*after care*" is seen as an important concept in the medical (cancer) domain then such visualization strategies can help to detect a defect. For this example, the reader is referred to Figures 5 and 8. In Figure 8, "*after care*" is only presented as an unimportant concept. In the textual summary it only appears in the description of *cancer case* but is not itself described. In a glossary representation it will not be among the first listed concepts. Hence, if this concept is important for the stakeholders they will be surprised on one hand but on the other hand they will get aware that something (i.e. a better description of *after care*) is missing.

VI. VISUALIZING THE PROGRESS AND SOURCE

Up to now visualization of model elements were described only. In fact a concept is not only related to another concept but it is also "related" to sources from which it was derived and it is related to a certain working state (progress). If the stakeholders need a holistic view of the model, then also their relationships to the sources and the working progress of the model is information that must be visualized appropriately. Figure 4 shows the relationships between a model element to its sources and its progress information.

In this section it will be firstly defined what is meant with source and progress. Then it will be explained how such information can be visualized.

A. Source

A source is any thing or media from which a model element like a concept can be derived.

In this approach three kinds of sources are distinguished

- natural languages requirement sentences,
- documents
- involved persons

A natural language requirement sentence is the smallest unit of source from which a model element (here a concept) can be derived. The requirement sentence itself can be selected from a document.

A document is any type of media in which a model element was found.

An involved person is any stakeholder who mentioned the model element.

Instead of using very small units of single requirements sentences only, this approach also allows to relate a model

element to the more coarse grained sources "document" and "involved person".

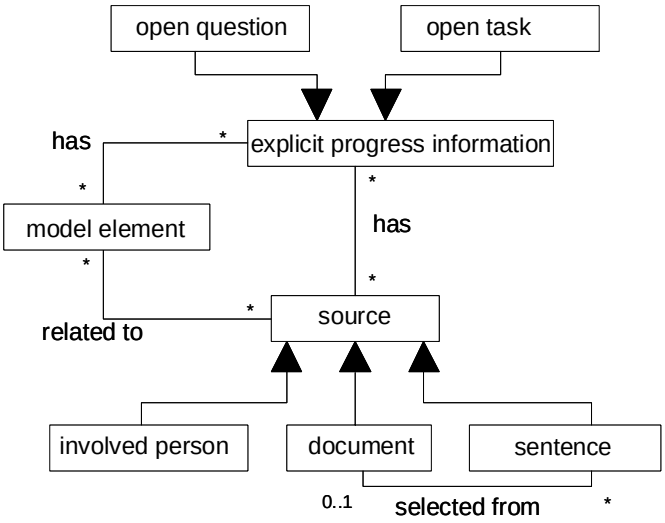

Figure 4: model element, progress and source

B. Progress

It might be surprising that progress, which is a certain state or snapshot of the modeling process, is specified in the meta-model. However "progress" can be divided into **explicit progress information** and **implicit progress information**. In this approach an explicit progress state is given if the designer declares that there are still some open questions or open tasks for a certain model element or a source. An implicit progress cannot be declared explicitly but is derived from the grade of completeness of the schema. Therefore the meta-model only specifies the explicit working progress. Nevertheless, the details of implicit progress information are also given here.

The **explicit progress information** (open questions and open tasks) is necessary for the following reasons:

- Conceptual modeling is always driven by decisions (i.e. decision to model certain information in a certain way, decisions to select and gather some information whereas other information is ignored etc.). Some of them can be made by the designer itself whereas other decisions need communication with end users. If the designer is not sure if he has made the right decision, then a possibility must exist to make a remark for asking the end user. Furthermore this remark must be related to the respective model element. Such a remark is not only a hint for the designer to ask somebody something, but also a concrete hint that the element is not yet finished.
- Not every task can be done at once. Some tasks must be done later. The open task remark helps the designer to remember these tasks (e.g., "*I must collected detailed information for the concept patient*"). Once again this remark is a concrete hint that something is not yet finished.

The **implicit progress information** can be derived by answering the following two questions:

- Is each column in the glossary view filled with a value?
- Is each concept related to at least one other concept and is the multiplicity information within each relationship specified for a certain concept.

The answers for the first question can be found inside the schema structure itself. In [22] a general method to calculate the progress was already introduced. Therefore this paper concentrates on the visualization part which was not the scope in [22]. The customization of the general method is only explained to the extend that is necessary to understand the visualization. Imagine a matrix similar to table 1 in the appendix. Each row describes a concept. Each column is reserved for a specific aspect of a concept (e.g., its examples, its definition, its value constraint, its quantity description – see Figure 3). A cell of a certain row and column can be empty or filled. The concept definition and the example column are two kind of information that must be filled out in order to be complete. The progress can be calculated by counting only those cells of columns which are filled with a value and dividing them by the total number of columns which must be filled out. If the total number of columns would be 2 (i.e. concept definition column and example column) and if only one is filled out for a certain concept at a certain point in time, then the progress state would be 50 %.

Additionally, the approach also considers the columns for value constraint and quantity descriptions. For the calculation of the progress state of these concept aspects, the general method described in [22] was customized and refined. The quantity description is not needed for every concept. Therefore it is optional. However, if a quantity (numeric value) is specified for a concept, then all the other information must also be specified (e.g., *average/maximum/minimum*, i*ncreasing per period or not*). The data type entry in the value constraint depends on the state of the concept. If the concept is already categorized as an attribute, then the data type must be filled out. If it is a class then it must not be filled out. If it is a concept which is not yet categorized to a class or an attribute and if the data type is empty, then the implicit progress information for that concept is defined as 0.5. This gives a hint that there might be still something missing.

Consider a concept which is a class. It has a quantity description and all the necessary information for this quantity description is specified. A concept definition is specified but no examples are given. In this case the progress state is 2 / 3 (~ 66 %). On the other hand if it is an attribute that has a data type entry but no (concept) definition and no examples then the state is 1 / 3 (~ 33 %). Finally if there is a concept which is not already categorized as a class or an attribute

and it has examples and a concept definition but no value constraint then the progress is calculated as 2 / 2.5 (~ 80 %).

The answer of the second question can be calculated by determining if a concept has at least one relationship to another concept. Then for every relationship which does not have predefined multiplicities (e.g., "*identifies*" relationship) the multiplicity information of the concept to its related concept is examined. The state of completeness for relationships and multiplicity information is defined as follows:

- If the concept has no relationship then the implicit progress information for the relationship progress state (RP) is 0. This means incomplete.
- If the concept has at least one relationship to other concepts, then the relationship progress state (RP) is determined by:

$$RP = \frac{\text{No. of specified multiplicities}}{\text{No. of relationships for a concept}}$$

For example, if the concept *cancer case* has 10 relationships to other concepts (e.g., *start location*, *histology description* etc.) but only for 4 of these relationships the multiplicities to the other concepts are defined, then RP = 0.4 (40 %).

The whole implicit progress information is calculated by building a sum of RP and the other progress state information (e.g., progress of example, concept definitions etc.). This is then divided by the possible number of progress information. The result is the overall progress state in percentage. As a continuation of the previous example, let us assume that cancer case would have an RP of 40 %. Furthermore it is categorized as a class and has examples and a concept definition. In this case the whole implicit progress state is 80 %. If all the relationships also have specified multiplicity information then the progress state is 100 %.

Implicit and explicit progress information is visualized separately, because situations can occur where a schema is already finished according to the implicit progress information, but it is not finished according to the explicit progress information. An example for such a situation is the following: The designer has already filled out and modeled the necessary information but in one case he is not quite sure if his design decision is correct. Since he has to ask one of the end users, he makes a note (open question) to ask somebody. In other words, from a structural point of view a certain model element in the schema is complete but it is not yet validated by the end user.

C. How to view the Progress information

There are several ways to **graphically** view the implicit progress information. One is to resize the concept. The more information about a concept reaches the state "complete", the bigger it appears in the graphical view. Alternatively the more a concept is not completed, the bigger it could appear in the graphical view. As a third possibility the concepts can appear in the different colors of a traffic light. The semantics of the colors are:

- *green*: concept is largely specified or even complete (>= 66 %);
- *yellow*: concept needs more information (>= 33 %)
- *red*: concept is barely specified (< 33 %)

It was decided to use this third possibility. For instance if the first alternative would have been chosen, then incomplete concepts only appear very small although the focus of the users attention should be directed to these incomplete concepts. On the other hand, if incomplete concepts are drawn very large then the mistake can occur that these concepts are seen as complete concepts.

Explicit progress information is graphically visualized in the same way:

- *red*: there is at least one open question or one open task respectively.
- *green*: no open questions; no open tasks.

Glossaries itself are a good view to visualize on a very detailed level that something is missing, since in this case the cell of a row and column is empty. If an overview of the progress is needed, then this can be achieved by a table consisting of four columns. The first column contains the concept names in each row. The second column contains the progress of this concept using a progress bar (see Figure 6). In the third column each cell is colored green if no open question is stored for a concept. If at least one open question exists, then the cell has a red color. In the fourth column the same visualization strategy is applied for open tasks.

A good strategy for verbalizing the progress was not found. Of course, there is always the possibility to verbalize the percentage of progress for each concept or to name the columns of a concept which are not filled out. However, glossaries or a graphical view are much better in such a situation since verbalization is a strategy which presents content itself and not the gaps.

D. How to view the relationship to the sources

In the same way as the amount of relationships to other concepts is visualized graphically, the strategy can be applied to visualize the relationship to sources. However, it cannot be concluded from such a counting strategy, that a concept with more relationships to sources is more important than a concept with fewer relationships. It might happen that a concept was only (but completely) found in one document or was specified by a single person. Hence, if in the graphical view a concept appears in a bigger size it only tells, that it has more relationships to different sources.

In the glossary view, the relationship can be viewed with the already well established strategy of a traceability matrix. In its most general form, there is one column for the concepts and columns for each kind of source (involved person, document and sentence). In the cell where a column and a row cross, a number indicates to how many sources of a

certain kind (e.g., document) a concept is related. Once again no adequate strategy was found actually for the verbalization view.

VII. THE TOOL

A. General Views

A tool (see Figure 7 in the appendix) was implemented to meet the requirement that verbalization, graphical representation and a glossary view must be combined.

The left upper part of the tool presented in Figure 7 is the verbalization view. Here the diagram appears as a description written in controlled language. These sentences are generated from the information of all the relationships and concepts. Relationship information include involved concept names, the name of the relationship (e.g., *is a*, *identifies*, *has*, *owns* etc.), and multiplicity constraints. Concept information is information about the value constraint, the format and the quantity description specified for a certain concept.

The right upper part of the tool is dedicated to the graphical representation. This is the classical form of representation used in conceptual modeling languages.

At the bottom the set of modeled concepts appear in a glossary style. The user has the advantage to use the list of concepts like a check list. He can look which columns are filled out and which are empty.

To ensure that the user will not be overburdened with three different views, of course it is possible for him to switch off one view completely. The user can also resize the different views to get a larger glossary view, a larger graphical view or a larger textual view.

Currently the model elements can only edited in the graphical view. The textual view offers only the possibility to insert controlled language sentences or read these sentences from a file. With a button in the text view panel, these sentences can then be transferred to the graphical view.

B. Visualization of Important Concepts

The visualization of important concepts currently is implemented in the following way. For the graphical view of important concepts, the designer has to click on the button with the "spyglass" icon. Then he gets a popup window with a menu of several possibilities. One option is to choose the visualization of important concepts. After he has selected this option, important concepts appear as described in three different sizes and colors (see Figure 8 in the appendix). If he wants to see a natural language summary of the important concepts, then he must select the tool menu option "Views" in the menu bar. Afterwards he must select the submenu item "Summary". Finally a window is popped up and displays the textual summary (see Figure 5). If he wants to see the glossary view then he also has to select the "Views" menu. Finally he must select the sub menu "Important concepts listing". A window is popped up where the concepts are ordered according to their importance.

C. Visualization of progress and sources

The graphical representation of the progress of concepts and their relationships to sources can be reached through the button with the "spyglass" icon in the graphic panel. The designer must then chose the corresponding option, depending of what he wants to see:

- Visualization of explicit and implicit progress information
- Visualization of relationships to sources

The right upper graphical part of the tool gives the required view as described (i.e. traffic light coloring paradigm for explicit and implicit progress information; three sizes and colors for concepts to visualize the number of relationships to sources).

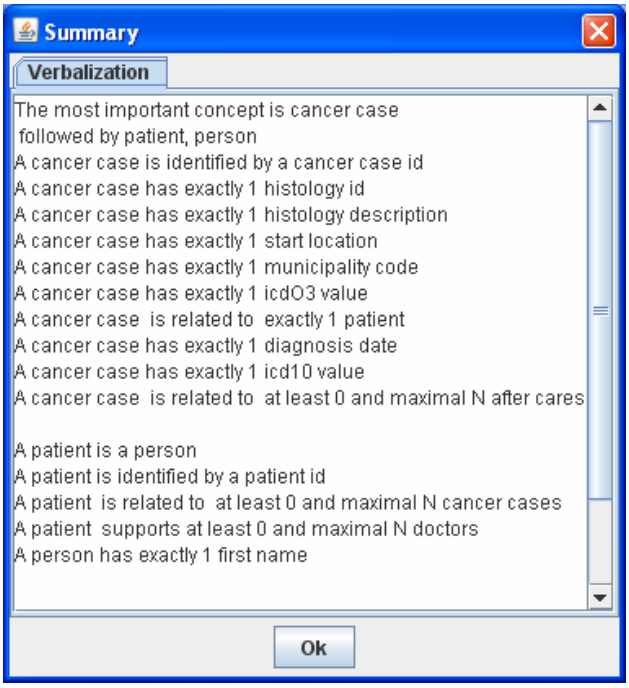

Figure 5: summary report

In order to get a glossary representation of progress information, the designer must navigate from the menu bar item "Views" to the sub menu "Progress information" and "Traceability overview" respectively. For each of the two options, a window is popped up which contains the necessary information (see Figure 6 for implicit and explicit progress information).

Verbalization strategies of progress information and sources are not supported at the moment.

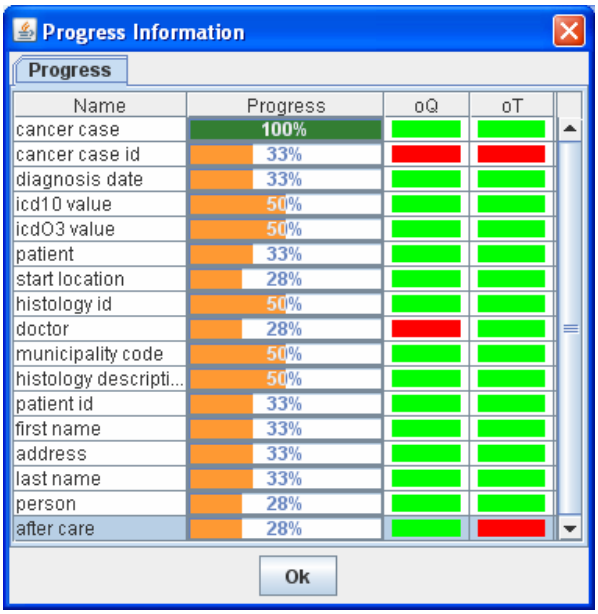

Figure 6: progress information

D. *Technical aspects*

The tool was implemented in Java and all the information of the concepts and their relationships are stored in a MySQL database.

The Model-View-Controller (MVC) architecture was used to manage the changes between the graphical and the glossary view. Inserting and updating of concepts and relationships is done in the graphical view. The user gets a property window for relationships and concepts. In these property pop up window he can insert and edit the details. Whenever details of a concept are changed then the graphical and glossary view is notified.

The verbalization is not trigged by these changes since the textual area in the left upper part of the tool (Figure 7) is also used as a simple editor for inserting a list of controlled English sentences which can then be transferred into the graphical and glossary representation. Instead a button in this area generates the verbalization from the model. The verbalization strategy itself is implemented within the MVC model classes for concept and relationship. Each of these classes has a public method "verbalize". Hence each object of these classes knows how to describe itself textually. The verbalization process itself is simply implemented by going through the entire concepts and relationships in a domain and by calling their method "verbalize".

VIII. CONCLUSION AND FUTURE WORK

It is very important that the result of conceptual modeling is negotiated with all the stakeholders. Since the stakeholders have different skills and knowledge background, different representation techniques should be used for the communication.

In this paper three representation techniques were combined to give all stakeholders the possibility to choose the most adequate one in a given situation. These visualization strategies were then applied to "structure" the schema between important and non important concepts in order to detect defects in the schema.

Since a schema must be seen in a greater context not only the visualization of the schema itself is relevant, but also the relationships of certain model elements to their sources as well as their actual progress of design. Both, overview of relationships to sources and progress information can help stakeholders to get a better picture about the current conceptual modeling state.

These strategies are based on previous research results, a survey of the literature and learning's made in projects.

In future, more special relationships might be added to this approach. Further special representation techniques for special purposes together with the existing techniques (i.e. progress or relationship to sources) should be studied for their optimal usability.

REFERENCES

[1] Ch. Kop, "Towards a Combination of Three Representation Techniques for Conceptual Data Modeling", First International Confernce on Advances in Databases, Knowledge, and Data Applications, 2009, pp.95-100.

[2] S. Ceri, (Ed.) *Methodology and Tools for Database Design*, North Holland Publ. Comp., 1983.

[3] A. Cockburn, *Writing Effective Use Cases.* Addison Wesley Publ. Comp., 2000.

[4] H. Dalianis, "A method for validating a conceptual model by natural language discourse generation". In P. Loucopoulos (Eds.), *Proceedngs of the Fourth International Conference CAiSE'92 on Advanced Information Systems Enginering*. Lecture Notes in Computer Sciences (LNCS) Vol. 594, Springer Verlag, pp. 425-444.

[5] M. Jose Escalona, G. Aragon, NDT. A Model-Driven Approach for Web Requirements IEEE Transactions on Software Engineering, Vol. 34, No. 3, 2008 pp. 377 - 390.

[6] N.E. Fuchs, S. Höfler, K. Kaljurand, F. Rinaldi and G. Schneider, "Attempto Controlled English: A Knowledge Representation Language Readable by Humans and Machines" In Norbert Eisinger N. and Maluszynski, J. (eds.): *Reasoning Web, First International Summer School 2005*, Lecture Notes in Computer Science (LNCS) Vol. 3564, Springer Verlag, 2005 pp. 213-250.

[7] X. Liang, and A. Ginige, "Smart Business Object - A New Approach to Model Business Objects for Web Applications", In *Proceedings of the first international Conference on Software and Data Technologies (ICSOFT 2006)*, Setúbal Portugal 2006, Springer Verlag, pp. 30-39.

[8] T. Halpin, 'UML Data Models from an ORM Perspective-Part 1', *Journal of Conceptual Modeling*, No. 1, 1998, www.orm.net.

[9] T. Halpin, A. Bloesch, "Data modeling in UML and ORM: a comparison", *Journal of Database Management*, 10 (4), 1999, 4 - 13.

[10] T. Halpin, M. Curland, "Automated Verbalization for ORM 2", In *Proceedings, OTM 2006 Workshops -On the Move to Meaningful Internet Systems 2006*, Lecture Notes in Computer Science (LNCS 4278), Springer Verlag, pp. 1181 – 1190.

[11] M. Jarrar, "Towards the notion of gloss, and the adoption of linguistic resources in formal ontology engineering" In *Proceedings of the*

15th International World Wide Web Conference (WWW2006). Edinburgh, Scotland, ACM Press, pp. 497-503.

[12] H.C. Mayr, C. Kop, "A User Centered Approach to Requirements Modeling", *Proc. Modellierung 2002*, Lecture Notes in Informatics LNI p-12, GI-Edition, 2002, pp. 75-86.

[13] D. Moody, "What Makes a Good Diagram? Improving the Cognitive Effectiveness of Diagrams in IS Development". In. G. Magyar, G. Knapp, W. Wojtkowski, W.G. Wojtkowski, J. Zupancic (Eds), Advances in Information Systems Development – New Methods and Practice for the Networked Society, Vol.2, Springer Verlag, 2007, pp. 481-492.

[14] G.M. Njjssen, T.A. Halpin, *Conceptual Schema and Relational Database Design – A fact oriented approach*. Prentice Hall Publ. Comp. 1989.

[15] J. Ryszard, D.L. Parnas, J. Zucker "Tabular Representations in Relational Documents", In Hoffman D., Weiss D.M. (Eds.) *Software Fundamentials – Collected Papers by David Parnas.* Addison Wesley Publishing Comp. 2001, pp. 71- 85.

[16] D.L. Moody, A., Flitman, "A Methodology for Clustering Entity Relationship Models – A Human Information Processing Approach", In. Proceedings of Conceptual Modeling (ER 1999), Lecture Notes in Computer Science (LNCS), Vol. 1728, 1999, Springer Verlag, Berlin, Heidelberg, pp. 114-130.

[17] D.L. Moody., "Entity Connectivity vs. Hierarchical Levelling as a Basis for Data Model Clustering: An Experimental Analysis" In DEXA 2003 Proceedings, Lecture Notes in Computer Science (LNCS), Vol. 2736, 2003, Springer Verlag, Berlin, Heidelberg, pp. 77-87.

[18] D. Bezerra, A. Costa, K. Okada, *SwTOI (Software Test Onotlogy Integrated) and its applicaton in Linux Test*. In Proceedings of the 3rd International Workshop on Ontology, Conceptualization for Information Systems, Software Engineering and Service Science, CEUR-WS, Vol 460, http://ftp.informatik.rwth-aachen.de/Publications/CEUR-WS/, pp. 25 – 36.

[19] N. Huang, Sh. Diao, "Structure-Based Ontology Evaluation" In IEEE International Conference on e-Business Engineering (ICEBE06), pp. 1- 6.

[20] R.J. Costello, D.-B. Liu, "Metrics for Requirements Engineeing, in Journal of Systems and Software, 1995, pp. 39 – 63.

[21] Ch. Pikalek „Messbare Qualität von Anforderungsdokument", Javamagazin, No. 1, 2006, pp. 75 – 81.

[22] Ch. Kop, "Work Progress Estimation from Structured Requirements Specifications", In Ch. Barry, K. Conboy, M. Lang, G. Wojtkowski, W. Wojtkowski (eds.). Information Systems Development, Springer Verlag, Vol. 2, 2009, pp. 909 – 922.

[23] Ch. Kop, "Conceptual modeling tool for novice designers", In International Journal of Metadata, Semantics and Ontologies, Vol. 3(2), April 2008, pp. 151 – 165.

[24] G. Kotoyna, I. Sommerville, *Requirements Engineering – Processes and Techniques*,Wiley Publ. Comp. 1998.

APPENDIX

TABLE 1 excerpt from the concept glossary

Concept name	Format	Datatype	DataSource	SrcFormat	SrcDatatype
cancer case			CCSTD		
cancer case id	L5	Number	CCSTD.ID	L5	Dezimal(5)
diagnosis date	YYYY/MM/DD	Date	CCSTD.DDATE	L10	CHAR(10)
icd10 value	L5	String			
icdO3 value	L5	String			
starting location		{left, right, unknown }			
histology id	L5	String			
histology description		Text			
patient			PSTD		
patient id	L6	Number	PSTD.PNUMBER	L5	Dezimal(5)
first name	L30	String	PSTD.FNAME	L21	CHAR(21)
last name	L30	String	PSTD.LNAME	L21	CHAR(21)
address	L255	String	PSTD.ADDRESS	L150	CHAR(150)
municipality code	L5	Number			
doctor					
person					

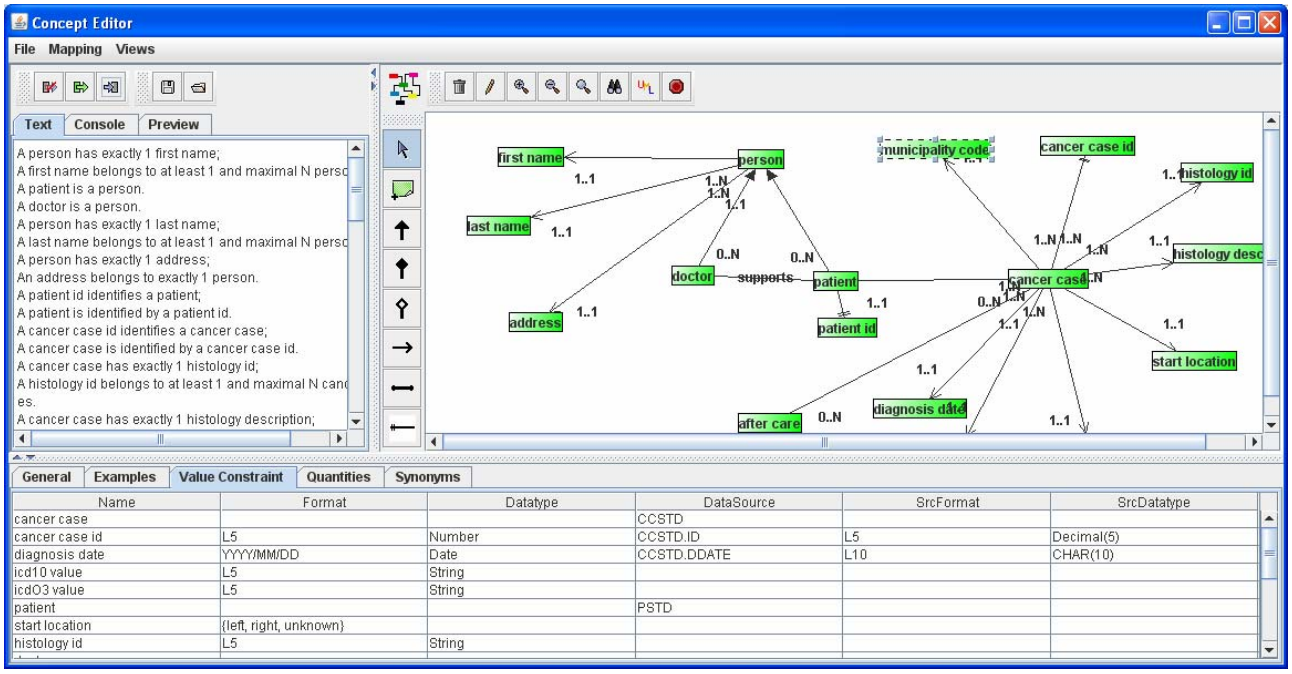

Figure 7: tool with the three presentation views

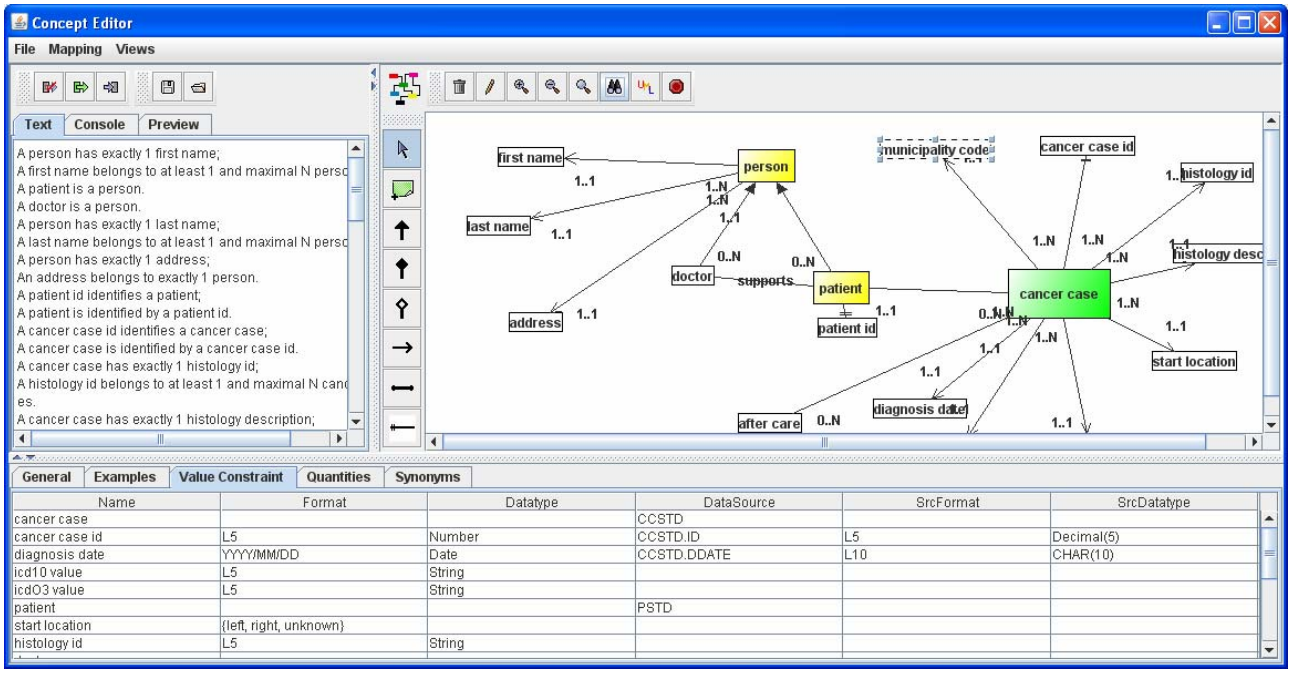

Figure 8: presentation of important concepts