Applying Similarity Vectors for the Selection of Reusable Schemas

Ana Paula AMBROSIO and Fouad GEORGES

Departamento de Estatística e Informática
Universidade Federal de Goiás - IMF
Campus II UFG - Caixa Postal 131
74001-970 Goiânia - GO - BRAZIL

e-mail: apaula@dei.ufg.br

Abstract: Schema Reuse can be defined as *the facility to construct a new system mainly from existing subschemas*. For this to be possible, the user must be able to select the best adapted modules easier and faster than if he should develop them from scratch. To help in this task, we have defined a Schema Construction System (SCS) that furnishes schema developers with an environment that promotes specification through reuse. The Retrieval Mechanism has been defined to support the user in his search for schematas liable to be reused in the development of new applications. In most software reuse systems, the retrieval of software components is based on semantic aspects. In the proposed mechanism, components selection is a flexible affair, where users can state their needs in a tolerant manner, permitting the system to search for components that are within a certain acceptable deviation from the conditions stated in the query. For this, vague query functions are offered by the retrieval mechanism, that take into account the need for flexibility not only at the semantic level, but at the structure level as well.

Keywords: *Reuse, Conceptual Modelling, Schema Retrieval, Similarity Vectors*

1. Introduction

Database schemas, also called conceptual schemas, correspond to formal representations of the Universe of Discourse as seen by information system's applications, and may be considered as a transient form, combining human understanding and machine representation. Schema contents thus determine the degree of visibility of the context's different elements, as well as their organisation and semantics.

The increasing size and complexity of conceptual schemas not only require modular modelling approaches, in order to permit evolutive construction, but also motivate the reuse of existing schemas for the construction of the respective modules. Reuse is thus intended to allow developers to benefit from already written and validated schemas (completeness, coherence, etc.) in the construction of new applications, all while conforming to the new design requirements. To help designers to reuse in the development process of database applications, we have defined a Schema Construction System (SCS) to introduce reuse within the Kheops environment.

As any reuse system, a powerful retrieval system had to be defined. Most reuse systems have chosen a semantic based approach to component retrieval [Bigg 89] [Reboot] [Ithaca]. We have opted for a mixed approach, giving users a greater flexibility. The system permits both a semantic and structural approach. This paper considers the retrieval aspects of schemata with respect to their structural characteristics. For this it builds upon the *similarity vectors* originally defined for the comparison of concept representations as to identify equivalences and conflicts for view integration purposes within the Kheops environment. This mechanism complements the Semantic Querying function which selects schema components based on their semantics.

The next section gives a brief overview of the Schema Construction System (SCS). This is followed by the presentation of the Retrieval Mechanism of the SCS of which the Structure Querying function is an integral part. Before attacking the specification of the Structure Querying function, the dictionary, which permits the retrieval of "similar" components, is described.

2. Schema Construction System

The importance and degree of information contained in the schemas has motivated the introduction of reuse into the Kheops System [Bouz 91], an environment defined to help developers of database applications by offering specification, validation and automatic generation procedures that guide the user in his quest. An analysis of the system showed that the introduction of reuse implied the modification of the specification methodology as to guide the user in the **construction** of an application mainly from existing elements as opposed to defining it directly from the design requirements. For this, a set of validated schemas, stored in a repository, is offered to the user as reuse options. This means acquisition tools that verify the quality of the schemas and that organise them as to permit their retrieval by flexible functions. The recovered schemas can then be modifed and integrated as to generate the global schema of the application. To undertake these functionalities, a Schema Construction System based on reuse has been defined [Ambr 95].

The modification and integration functionalities were already offered by the Kheops

environment. The main difficulty met during the exploitation of existing schema definitions was to discover the suitable elements liable to satisfy the design requirements. This functionality is offered by the Retrieval Mechanism which is based on semantic and structural aspects of the schemas.

Semantic component retrieval is done by means of a semantic description of the sub-schema contents attached to the schema in the form of a tag, and a flexible retrieval mechanism that permits the recovery of "similar" schemas. This description aims to capture the UoD portion represented in a given sub-schema, as well as the designer's viewpoint while perceiving this reality. This permits the system to abstract from labelling and structure differences that may exist between the user's mental model of the application, and the vision represented in the component. The tag contains information concerning the environment (setting and functional-area) for which the schema was developed and the main key-concepts contained in the schema. The use of a descriptor for the schema permits an intial selection based on the abstraction of the schema and thus relieves the user from the need for a close analysis of the schema contents. The query specification is made using a select statement that offers the user the possibility of stating modification criteria in order to select not only exact matches, but similar solutions as well. Automatic query modification is applied by traversing the semantic relations contained in the dictionary [Ambr 95].

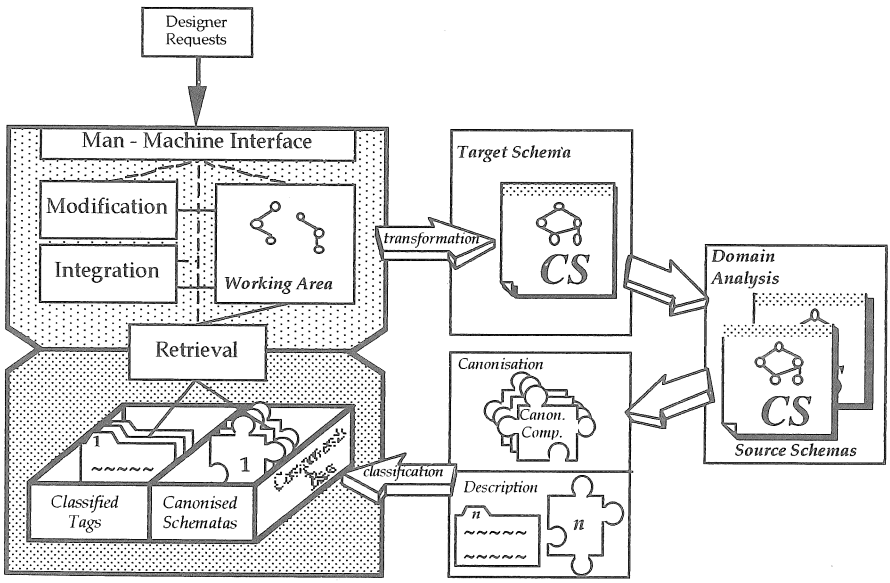


Figure 1: Schema Construction System

Semantic retrieval permits to filter the desired components according to the semantics they portray. However, when several components are recovered, the user must select the one(s) that are best adapted to the applications requirements. For this the Retrieval Mechanism offers the Structure Querying Function as to recover those that are best adapted to a situation already in place or that are a better representation of the mental model the user has of the application. The Structure Retrieval

function searches for schemas based on their structure by comparing a skeleton schema furnished by the user to the schemas in the components. This is done using similarity vectors [Comy 90], which have been extended and adapted for reuse. In the following this functionality is discussed, but before, we present the Kheops system's dictionary which is used by the Semantic and Structure Retrieval Functions to modify user queries and to search for similar concepts to those stated in the request.

3. The Dictionary

The dictionary is divided in two complamentary dictionaries containing general syntactic and semantic information [Meun 95]. In the syntax dictionary, structural characteristics are represented. They include regular and irregular verbs with their conjugation in the present tense, simple and composed nouns with their canonical form and genre (singular, plural, masculine, feminine), adjectives in their feminine and masculine forms, prepositions and abbreviations. Exceptions are also recorded. These include verbs whose root is not the same in all tenses, nouns whose plural is not obtained by the addition of 's', and adjectives whose masculine and feminine are not regular. All word entry in this dictionary makes reference to the concept it is associated to in the semantic dictionary.

Semantic knowledge is stored using semantic cases, canonical graphs of Sowa and semantic relations. Semantic relations establish links between word meanings (concepts). They include links of synonymy, antonymy, hyponymy/hypernymy (is-a), meronymy/ holonymy (part-of) and similarity. Semantic cases were introduced by Fillmore [Fill 68], and later expanded and used by Sowa [Sowa 83]. They usually express the role played by each noun in a sentence (e.g. in the sentence "Workers manufacture shoes with leather", "Workers" has the semantic case "agent", "shoes" has the semantic case "result" and "leather" plays the role of "material"). These cases are more semantic than the usual syntactic cases (subject, direct object, adverbial phrase of place, etc.) in that they are not dependent upon the surface structure of the sentence (i.e. the syntactic construction of the sentence).

The canonical graphs of Sowa [Sowa 93][Way 92] are the structures that specify on one hand the relationships expected between concepts, i.e. the "semantic cases" of Fillmore (called "conceptual relations" by Sowa) and on the other hand the "semantic constraints" of the expected concept (also called "selectional constraints" by Sowa). Examples of semantic cases are "agent", "object" and "location", while examples of semantic constraints defined for the preceding cases are "Human", "Physical-object" and "Place". This means that an agent in a sentence must be a human being, while the object must be a physical-object and a location must be a place.

4. Structure Querying Function

The Structure Querying Functionality offers the designer the possibility of recovering components by furnishing a vague description of the desired concepts structure. In fact, structure querying is offered as a complementary function for the construction of conceptual schemas. To search all the component base for a given structure can become very expensive and time

consuming. We therefore propose its use when a system structure is already in place, and the components being searched for must conform to it or as a selection criteria. In this case, it becomes in fact a choice criteria when several components have already been recovered through semantic querying or browsing. In this manner, the structure search is limited to the already pre-selected components.

Schema selection is based on a query skeleton schema (a schema that gives an overall and vague representation of the desired structure). In this manner, schemas from the repository can be recovered when they include entirely or partially the skeleton schema given by the user. In fact, if we see the skeleton schema as one view and the schema from the repository to which it is compared as the other view, the processes of view integration and recovering schemas for reuse (based on their structure) resemble in many points. In both cases, a pairwise comparison of the concepts in the views must be undertaken to verify their similarity. Based on this calculus, it should be possible to decide if they represent the same concept in the real world or are inclusive. When an acceptable degree of similarity is found between two concepts, they can be integrated, or in our case, reused.

For structure querying we have extended the similarity vectors used by the view integration and evolution mechanisms offered by Kheops [Bouz 90][Keda 95]. The choice of this comparison criteria was greatly influenced by the environment for which the system was developed and by the good results obtained by the method in the other mechanisms. We believe however that other view integration methods may also be adapted to the retrieval of reusable components since most them also follow structure comparison between schemas. In the following we describe how similarity vectors are defined and how they have been modified to be applied in reuse.

5. Similarity Vectors

Similarity vectors permit the comparison of two conceptual schema structures (attributes, entities and relationships and schemas) to judge their degree of similarity. In fact, the similarity measure calculated can represent the global degree of similarity or inclusion that exists between two structures. The evaluation of a similarity vector is progressively made by a successive evaluation of each of its components. The values of these components are calculated by arbitrary functions which correspond to heuristics and can be refined or parametrized depending on the application problem. In this manner, greater importance can be given to certain vector components as compared to others, for example, a greater weight can be give to the name as compared to the structure. Furthermore, the system permits the specification of a similarity threshold which establishes the minimum degree of similarity that must be satisfied to consider two concepts as candidates for integration.

Similarity Vectors that calculate the similarity between object components have been implemented using a method based on *semantic unification* as proposed in [Comy 90]. Semantic unification is based on three notions: equivalence, similarity and dissimilarity. The equivalence between two concepts may be considered as a similarity with the highest degree. Two concepts are dissimilar if they are neither equivalent nor similar. Pairs of concepts of the same type from each structure group can be compared by a set of properties which define a similarity degree whose value depends on the nature and the number of common properties. Basically the comparison is

based on the concept's label and structure. The user has the possibility of defining the degree of importance of the different comparison aspects in the similarity calculus. This is done by attributing weights for each aspect. Intuitively the semantic measures for the three structures can be described as:

• *Semantic Equivalence* between

• attributes - Two attributes are considered equivalent if they have same label and same domain.
• entities -Two entities are considered equivalent if they have same label and are composed of equivalent attributes.
• relations - Two relationships are considered equivalent if they have same label and are composed of equivalent entities and attributes.
• schemas - Two schemas are considered equivalent if they are composed of equivalent relationships, entities and attributes.

• *Semantic Similarity* between two concepts (attributes, entities, relations and schemas) occurs if they have the same or similar labels, or the same domain, or a similar structure. The degree of similarity may range from 0 to 1. Being the equivalence and the dissimilarity the extremes (i.e. equivalence = similarity 1, dissimilarity = similarity 0).

• *Semantic Dissimilarity* between two concepts occurs when they are neither equivalent nor similar.

To diminish the number of comparisons when treating relations, the role played (as defined by Sowa in his graphs) by the concept (entity) is taken into account [Meta 93]. Comparisons are made only between entities that play the same role. For example, if we have the associations "a client buys a car from a garage" and "a garage buys a car from a client", there would be a tendency to unify both clients and both garages. However, these entities play different roles in each association. Therefore, before making comparisons the role each entity plays is taken into account, and only those that play the same role are compared.

5.1. Similarity between Attributes

Two attributes are considered equivalent if they have the same label and the same domain. If they are not equivalent but relations can be established between the label and domain values, the attributes are said to be similar. If the attributes are niether equivalent nor similar, the attributes are said to be dissimilar. Weights can be attributed to the label (wl) and domain (wd) aspects. The weight defines the degree of importance of said aspect in the similarity calculus. The sum of both weights must be 1 (wl+wd=1). The result of this comparison is represented in a similarity vector SIM_{att}(label, domain).

After values have been given to both components of the attribute similarity vector, an overall similarity value can be calculated for the vector through the weighted sum of the component values (SIM_{att}(l, d) = (wl*l+wd*d)). The degree of similarity may range from 0 to 1. The comparison procedure for attributes is described in the following.

- **Label Comparison**

When comparing labels, the system makes use of the concept graphs contained in the dictionary (cf. section 3). Semantic similarity between two concepts is defined based on their relative position in the concept graphs of the dictionary [Keda 95]. They are said to have a:

- *Very Strong Link*, if they represent the same concept (synonyms). In this case, the similarity value for the label component of the similarity vector is 1.

- *Strong Direct Link*, if they are directly related in the hierarchy (a concept can be reached from the other by traversing a semantic link in the graph). In this case, the similarity value for the label component of the similarity vector is 0,75.

- *Strong Indirect Link*, if they are indirectly related in the hierarchy (a concept can be reached from the other through a path in the graph following a unique branch). An example of this type of link is the link that exists between the concept Person and Teacher, if there is an intermediary concept Employee which cuts the path between the concepts in two. In this case, the similarity value for the label component of the similarity vector is 0,6.

- *Relatively Strong Link*, if the two concepts are directly linked to a common supertype. In this case, the similarity value for the label component of the similarity vector is 0,5.

- *Weak Link*, if the two concepts are indirectly linked to a common supertype. In this case, the similarity value for the label component of the similarity vector is 0,25.

- *No Link*, if there is no path between the two concepts except one passing through the hierachy root. In this case, the similarity value for the label component of the similarity vector is 0.

- **Domain Comparison**

When comparing domains, four possible situations can occur:

- *The domains are identical.* In this case, the similarity value for the domain component of the similarity vector is 1.

- *One domain is included in the other.* In this case, the similarity value for the domain component of the similarity vector is 0,75.

- *The domains are discrete and intersecting.* In this case, the similarity value for the domain component of the similarity vector is the result of the calculation of the equation [number of intersecting domain values]/[total number of domain values].

- *The domains are continuous and intersecting.* In this case, the similarity value for the domain component of the similarity vector is the result of the calculation of the equation [length of intervals

intersection]/[length of intervals union].

Examples:

Suppose wl= 0.8, wd= 0.2

att1: age: [20-70]	label comparison: age = age -> l = 1
att2: age: [0-120]	domain comparison: [20-70] c [0-120] -> d=0.75
	$SIM_{att}(0.8*1, 0.2*0.75) = \underline{0.95}$ (similar)
att1: name	label comparison: name = name -> l = 1
att2: name: text	domain comparison: empty, text -> d=1
	$SIM_{att}(0.8*1, 0.2*1) = \underline{1}$ (equivalent)

OBS: When the domain of an attribute is not given (empty), the similarity with any other domain is equal to 1.

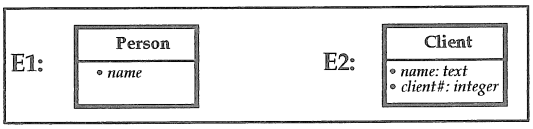
5.2. Similarity between Entities

Two entities are considered equivalent if they have the same label and are composed of equivalent attributes. Notice here that similarity relations between structures are recursive. The definition of equivalence, similarity and dissimilarity of entities are given in terms of equivalence, similarity and dissimilarity of attributes. Analogously, if the entities are not equivalent but relations can be established between their label and structures, the entities are said to be similar. If the entities are niether equivalent nor similar, they are said to be dissimilar. The result of this comparison is represented in a similarity vector SIM_{ent}(label, struct). Here also, weights can be attributed to the label (wl) and structure (ws) aspects. The sum of both weights must be 1 (wl+ws=1).

After values have been given to both components of the entity similarity vector and to the weights, an overall similarity value can be calculated for the vector through the weighted sum of the component values ($SIM_{ent}(l, s) = (wl*l+ws*s)$). The degree of similarity may range from 0 to 1. The comparison procedure for entities, as for attributes, is divided in two parts. The calculus of label similarity which is the same as for attributes, and the calculus of structure similarity which is described in the following. The comparison of entity structures is based on the set of attribute similarity values obtained for the attributes that compose the entities. Suppose an entity E1, with n attributes, and an entity E2 with m attributes. The set of attribute similarity values is obtained by comparing each attribute in one entity to the attributes of the other entity (m x n comparisons). The best result for each attribute comparison is retained.

To calculate the inclusion of an entity structure in the other entity's structure, the sum of the attribute similarity values retained is divided by the total number of attributes in the entity whose inclusion is being calculated.

Example:



Suppose wl: 0.5, ws: 0.5
label comparison: Person = Client suppose Client is-a Person ->
strong direct link -> l = 0.75
structure comparison:
att1: name $SIM_{att}(1, 1) = (2)/2 = 1$
att2: name: text

att1: name $SIM_{att}(0, 1) = (1)/2 = 0.5$
att2: client#: integer
best attribute similarity for name: 1

s = Incl12: (Σ att similarity)/(att in E1) = (1)/1 = 1

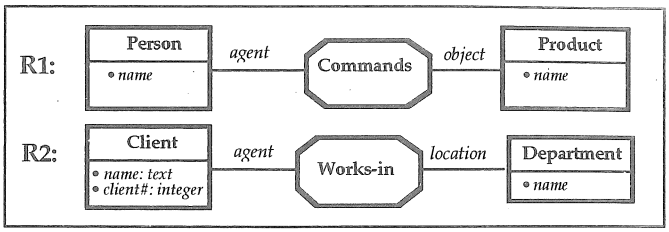
$SIM_{ent}(l, s) = (wl*l+ws*s) = (0.5*0.75 + 0.5*1) = 0.88$

5.3. Similarity between Relationships

Two relationships are considered equivalent if they have the same label and are composed of equivalent entities and attributes. As for entities, the definition of equivalence, similarity and dissimilarity is recursive. If the relationships are not equivalent but relations can be established between their label and structures, the relationships are said to be similar. If they are niether equivalent nor similar, they are said to be dissimilar. The result of this comparison is represented in a similarity vector SIM_{rel}(label, ent_struct, att_struct). For relationsips weights can be attributed to the label (wl), to the entities (we) and to the relationship attributes (wa) aspects. The sum of the three weights must be 1 (wl+we+wa=1). After values have been given to the components of the relationship similarity vector, an overall similarity value can be calculated for the vector through the weighted sum of the component values ($SIM_{rel}(l, e, a) = (wl*l+we*e+wa*a)$). Here also, the degree of similarity may range from 0 to 1. The calculus of label similarity is the same as for attributes and entities. The relationship structure similarity is described in the following.

The comparison of relationship structures, is based on the set of attribute similarity values and entity similarity values obtained for the attributes of the relationship and the entities that participate in the relationship. However, to diminish the number of comparisons, similarity vectors verify first the role of the entities in the relationship [Meta 93]. Only those entities that have the same role in the relationship are compared. If an entity does not have a corresponding entity with the same role, the degree of similarity for that entity is 0. If no role is attributed to an entity, a Cartesian product of said entity with all the entities of the other relationship is made. The overall best similarity is retained for the entities. The ent_struct value is calculated by the weighted average of the retained entity similarities. The att_struct is calculated analogously to the entity structure.

Example:



label comparison: Commands = Works-in -> l = 0
structure comparison:

E1: Person $SIM_{ent}(l, s) = (0.75 + 0.75)/2 = \underline{0.75}$
E2: Client
Product entity in R1 does not have a corresponding entity (with same role) in R2
$SIM_{ent}(l, s) = \underline{0}$
Overall similarity for entities = $(0.75+0)/2 = \underline{0.38}$
attribute similarity = 1 since no attribute was defined in Commands of R1.

$SIM_{rel}(l, e, a) = (wl*l+we*e+wa*a) = (0.5*0 + 0.3*0.38 + 0.2*1) = \underline{0.31}$

5.4. Similarity between Schemas

Similarity vectors also permit the calculation of the degree of inclusion of a skeleton schema (SkSch) in a repository schema (RepSch). SkSch can be defined using attributes, entities and relations. However, since they are supposed to give an idea of the desired structure, they can be vague and incomplete, e.g., they may state entities with just the name or a relation with only one entity or with a partial structure.

The similarity vector for schemas is defined based on the relationship, entity and attribute similarities. The label similarity is not used since it is not relevant to the degree of inclusion of the SkSch structure in the RepSch. Thus, SIM_{sch}(rel_struct, ent_struct, att_struct). To maintain a coherent approach with that already adopted, schema comparison starts by relations, followed by entities and finally by attributes. This permits the treatment of the more complex structures first.

Weights can be attributed to the importance of the structures in the similarity calculus. For SkSch, weights can be attributed to the relationship (wr), to the entities (we) and to the attributes (wa) aspects. The sum of the three weights must be 1 (wr+we+wa=1). After values have been given to the components of the schema similarity vector, an overall similarity value can be calculated for the vector through the weighted sum of the component values ($SIM_{sch}(r, e, a)$ = (wr*r+we*e+wa*a)).

We would like to observe that the weight to be attributed to the different aspects should take into account the structures used in the definition of the skeleton schemas. If the SkSch has only one relationship, the weight attributed to the relationship similarity should be 1 since it is in fact the degree of inclusion of the given relationship in the RepSch that determines the degree of inclusion of the SkSch in the RepSch.

Schema inclusion verification starts by comparing each relationship in the SkSch to every relationship in the RepSch. Inclusion calculation is maintained the same as for the original similarity vectors. This means that the entities participating in the SkSch relation and the relation attributes are also compared to the entities and attributes of the RepSch. If certain concept compontents have not been specified in the SkSch, they are considered as included with a degree of 1. For example, if no attributes have been defined for the SkSch relation, and the RepSch relation has a list of attributes, the degree of inclusion of an empty list of attributes in a list with elements is defined as 1 by the similarity operators. After each relation in the SkSch has been compared to the relations in the RepSch, the best similarity measure obtained for each relationship is retained. The degree of inclusion of the relationships of the SkSch in the RepSch, is calculated by the average of the retained similarities.

Next, each entity in SkSch that does not participate in a relationship (those have already been compared), is compared to the entities in the RepSch that do not participate in a relationship. If no other entity exists, the degree of similarity is set to 1. Entity similarity calculation is recursive, i.e., the attributes of the entities are also used in the comparison. If no attribute has been defined for the entities, the degree of structure similarity is 1. The best similarity measure obtained for each entity is retained. The degree of inclusion of the entities of the SkSch in the RepSch, is calculted by the average of the retained similarities.

The same approach is adopted for the attributes. If no other attributes exists, the degree of similarity is set to 1. If incomplete attribute specifications are made that include only the name and not the domain, the degree of similarity for the domain component of the similarity vector is set to 1. The best similarity measure for the attribute is retained. The degree of inclusion of the attributes of the SkSch in the RepSch, is calculted by the average of the retained similarities.Finally the weighted sum of the similarity measures obtained for the attributes, entities and relationships is calculated. This will give the degree of inclusion of the SkSch in the RepSch.

6. Conclusion

Software reuse has for long been identified as a requirement for the efficient development of information processing systems. For many researchers, reuse is the only envisaged solution to the cost-effectiveness problem in the development of large software systems. Conceptual schemas, as part of a software system, share many of its design concepts and problems. Meanwhile, in the reuse context, conceptual modelling is particularly distinguished by its semantic requirements. The need to treat the semantic aspects of conceptual schemas has already been identified by several studies concerning the database development activities, such as view integration and application specification.

The proposed Structure Querying function, as part of the schema retrieval mechanim, treats one of the main bottlenecks of a reuse-based approach for the construction of conceptual schemas. Since the selection criteria is applied after a first filtering of components based on the portion of the UoD they represent and the semantic they portray, the high cost it implies is drastically diminished. Furthermore, the selection of schemas based on an outline of the desired structure responds to the need users have of describing the mental model they have of the application, all while offering a

certain flexibility which permits the retrieval of components that are similar to the stated requirements.

This flexibility is the more interesting in the domain of conceptual schemas since, as opposed to software, they are easily modified and integrated. Much research in this sense has been done which has lead to the development of CASE tools that guide/help the user in the modification/integration process (e.g. Kheops). To these tools, the introduction of reuse means the development of a schema retrieval mechanism which permits the selection of components in the design process. The proposed mechanism, due to its modularity, can be easily adapted to perform this task.

Acknowledgments: I would like to thank Ms. Elisabeth Metais and Zoubida Kedad for their patient and informative discussion of the above subject.

Bibliography

[Ambr 95] A.P. Ambrosio, *A SEMANTIC QUERY MECHANISM FOR THE RETRIEVAL OF REUSABLE COMPONENTS: Application to Database Schema Reuse*, PhD Thesis, Univ. Paris VI, June 1995.

[Bigg 89] Biggerstaff and Perlis (Editors). *Software Reusability (vol 1 and 2)*, Addison-Wesley, 1989.

[Bouz 90] M. Bouzeghoub and I. Comyn-Wattiau, *View Integration by Semantic Unification and Transformation of Data Structures*. 9th International Conference on E-R Approach, Lausanne CH, Oct 1990.

[Bouz 91] M.Bouzeghoub and E.Metais, *Semantic Modelling of Object Oriented Databases*, Proceed. of VLDB91, Barcelona Spain, Sep 1991.

[Comy 90] I. Comyn-Wattiau, *L'intégration de vues dans le système expert SECSI*, Thèse de Doctorat, Univ. PARIS VI, Oct 1990.

[Fill 68] C. Fillmore, *The case for case*, Universal in Linguistic Theory, Bach & Harms, Holt, Rinehart and Winston, New York, 1968.

[Ithaca] **Ithaca Project**, Reports available through anonymous FTP on *//cui.unige/OO-articles/ITHACA* or on *//ithaca.elet.polimi.it*

[Keda 95] Z. Kedad, *Aspects Linguistics dans l'Integration de Vue de Kheops*, Rapport de Stage DEA, Lab. PRISM, Versailles, 1995.

[Levr 94] G. Levreau and M. Bouzeghoub, *HypER: An Extended E/R Model with Hypertext Facilities*, WITS'94, Canada, Dec 1994.

[Meta 93] E. Métais, J-N Meunier and G. Levreau, *Database Schema Design: A Perspective from Natural Language Techniques to Validation and View Integration*, XII International Conf. on E/R APPROACH, Dallas, Dec 1993.

[Meun 95] J.N. Meunier, *Linguistic Aspects in Conceptual Modelling*, personal notes.

[Reboot] **Reboot Project**, Reports available through anonymous FTP on *//ftp.idt.unit.no/pub/reboot*

[Sowa 83] J. F. Sowa, *Conceptual Structures*, Ed. Addison-Wesley, 1983.

[Way 92] E.C. Way, *Conceptual Graph Overview*, JETAI, no. 04, 1992.