A Toolkit for Reuse in Conceptual Modelling

Raul Ruggia
Instituto de Computación
Universidad de la República
Uruguay
ruggia@fing.edu.uy

Ana Paula Ambrosio
Departamento de Estatísitca e Informática
Universidade Federal de Goiás
Brazil
apaula@dei.ufg.br

Abstract:

This paper proposes a toolkit for applying Reuse in Conceptual Modelling. The main objective is to cope with the problems of complexity in the Conceptual Modelling activity. In a long-term perspective this proposition intends to settle the basis for a larger application of Reuse in Information System development. While research in Software Reuse has revealed that the application of Reuse in software development is extremely difficult, Conceptual Modelling appears as a more promising area because it manipulates simpler objects: conceptual schemas.

The proposed toolkit provides reuse-oriented services to KHEOPS database design environment. These services include: quality validation of reusable components, component selection from the Repository, and new conceptual schema construction by customising and composing reusable ones. Reusable components consist of an Extended Entity-Relationship schema as well as other information like executable reuse guidelines.

1. Introduction.

Conceptual Modelling is a complex process highly driven by the designer and involves capturing large amounts of problem and domain knowledge. In order to cope with this complexity as well as to enhance the quality of the resulting conceptual schemas different techniques have been proposed. Some of them are: the acquisition of application knowledge from natural language [Kers 86][Meta 93] and form descriptions [Choo 88], and expert systems for guiding the construction of the conceptual schema [Bouz 85]. These techniques improve productivity and quality but they always begin the design from scratch with a very low profit from previous design work.

A new emerging approach for enhancing productivity and quality in Conceptual Modelling consists in reusing existing conceptual schemas in the development of new ones. Reusing existing certified (i.e., with validated quality) conceptual schemas would reduce the difficulties in acquiring and specifying real world concepts in terms of a conceptual model as well as reduce the efforts of validating the conceptual schema.

Reuse has been widely applied in other Computer Science domains where it has been shown to contribute to the reduction of cost-effectiveness problems in software development [Bigg 89a]. In the context of Conceptual Modelling, although it has been recognised as a promising design strategy, Reuse has been little explored [Louc 92a][Roll 92]. Existing knowledge in the area includes proposals for constructing and reusing generic entity types through instanciation and specialisation [Cast 92][Seo 94][Delc 96], and proposals of classification and component recovery methods [Ambr 95][Li 96].

A reuse-based technology needs to cope with different technical issues: *Component Representation*, which concerns the specification of the component's logical structure (i.e., what is represented in a reusable component) as well as the storage support; *Component Classification,* which concerns the adequate organisation of the components in the Repository with the aim of enabling an effective component retrieval, *Component Certification,* which concerns the control of quality properties to be satisfied by the reusable components in order to achieve re-user's confidence, *Component Selection,* which enables the efficient and flexible retrieval of components from the Repository. *Component Customisation,* which enables to adapt retrieved components to the new application. *Component Composition,* which enables to integrate reusable components to form the new application by connecting them.

This paper proposes a toolkit that provides reuse-oriented services to the KHEOPS database design environment [Bouz 91][Dela 95]. KHEOPS is a database design environment that provides different types of interfaces for specifying Extended Entity-Relationship (EER) schemas: graphical, natural language, and declarative interfaces. KHEOPS also provides view integration functionalities, generating the logical schema and DBMS code (C-SQL, C++, CO2) from the conceptual schema specification. By applying the reuse-oriented services, the designer develops new conceptual schemas by reusing other existing ones.

The main contributions of this work concern the specification of a toolkit which enables to apply a reuse-based methodology in Conceptual Modelling as well as technical aspects concerned with the proposed mechanisms for component certification, selection, customisation and composition. More concretely, this work proposes an unified framework for conceptual schema quality validation, it proposes a component Selection mechanism based on semantic query modification techniques, and an operators-based approach to schema manipulationt. The tools which provide selection, customisation and composition functionalities have been implemented, and a first version of the Quality Control Tool is currently being implemented.

In this paper we focus on three main technical issues treated by the toolkit: (i) the quality validation mechanism applied to certify the reusable components; (ii) the component selection mechanism, which enables to retrieve schemas through vague queries; and (iii) the component customisation and composition operations, which enable to assemble the reusable schemas in order to build a new one.

Due to limits in paper extension, technical descriptions are not so detailed as desired. More detailed specifications can be found in [Ambr 95a] and [Rugg 96].

This paper is structured as follows: Section 2 presents the general aspects of the adopted approach to Reuse in Conceptual Modelling. Section 3 presents the tools in the toolkit. Section 4 presents the conclusion.

2. The approach to Reuse in Conceptual Modelling.

The application of Reuse techniques in software development has been widely studied [Bigg 89][Krue 92][Mili 95][Thun 92][Cons 95]. In this work we reused much of this knowledge to apply Reuse in the context of Conceptual Modelling.

In this work we adopted a compositional reuse approach (where software development is performed through the *composition* of existing software components). In this approach, the reusable components are mainly based on Extended Entity-Relationship (EER) schemas [Bouz 94].

The *development with reuse* process (the development of new applications by reusing existing components) begins with the selection of reusable components which EER schema represent certain real world concepts. After this, the EER schemas in the retrieved components are customised, so as to adapt them to the new requirements, and composed to build a new conceptual schema. The *development for reuse* (the development of new reusable components) consists of the construction of the reusable components, followed by a quality validation step and their insertion in the Repository.

The reuse-oriented services provided by the proposed toolkit to KHEOPS include: *Certification* of reusable components, *Insertion* of new components in a Repository, *Selection* of components through a flexible query mechanism, *Customisation* and *Composition* of the retrieved components through schema operators, and *Assistance* to users through reuse-guidelines and automatic schema transformations. From a methodological point of view, Reuse is integrated with KHEOPS methodology as a source of conceptual schemas (Figure 1).

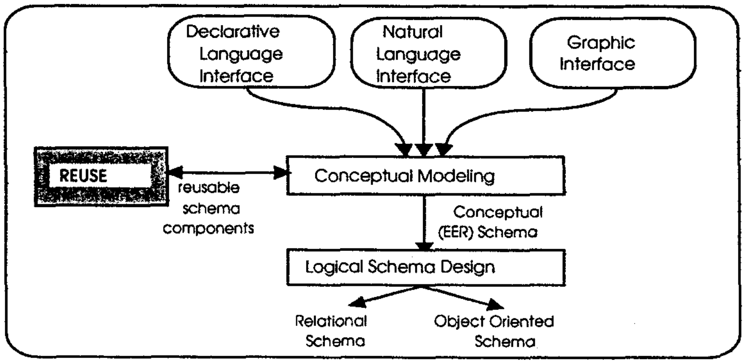


Figure 1. Integrating Reuse into KHEOPS methodology.

The Reuse Environment.

The different objects and resources involved in the proposed toolkit constitute the so-called Reuse Environment. Briefly, the Reuse Environment consists of: a Linguistic Knowledge Base (*LKB*) which stores the Domain Models [Prie 91]; the Repository which includes the reusable components and the Faceted Index; and the Certification Framework which contains the quality properties, property classes and certification instances to be applied to the components.

The *LKB* provides linguistic information to different modelling operations in KHEOPS: the natural language interface, the schema paraphrasing and the view integration operations [Ambr 95]. It is intended to be either specific to applications, to application domains or a general purpose semantic dictionary.

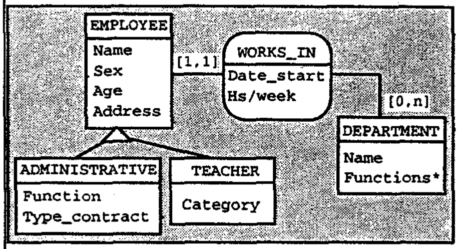
In the proposed Reuse Environment, the unit of reusable knowledge is the *reusable schema component*. A *reusable schema component* includes not only real world objects but also additional information about how to exploit these representations. The knowledge concerning the real world concepts are represented using an Entity-Relationship schema, references to Domain Model objects and textual documentation. The references to the Domain Model enable to fill the gap between the software artefact and the real world concepts. The textual documentation provides context information about the represented real world situation. Other parts of the reusable knowledge consist of information about the EER schema to be reused: a summary of the quality properties satisfied by the schema and statements about the ways of reusing the schema (i.e., reuse guidelines).

The references from components to Domain Model objects are stated in the *Semantic Descriptors*. A *Semantic Descriptor* is a triple *<Key-Concepts, Setting, Functional-Area>* which describes the meaning of the reusable component in terms of Domain Model concepts. Using these three descriptors, a reusable component is interpreted as used in the *Setting* to represent the *Key-Concepts* for the purpose of offering the functionalities in the *Functional-Area*. The Semantic Descriptors also serve as basis to define a Faceted Index [Prie 87] on the set of components. The Faceted Index is a component classification mechanism which enables to retrieve the selected components efficiently.

Example: *A reusable schema component.*

This example shows a component named `Personal-Administration` intended to be used in a *University* context. This component represents the *Employee* and *Works* real world concepts to perform *Employee Administration* functions (Figure 2). More precisely, the `key-concept` Employee is connected with the LKB semantic entry Employee-Person, and it is represented in the `schema` through the entity type `EMPLOYEE`.

```
name: Personal-Administration
semantic-descriptor:
    ({(Employee,
              Employee-Person,
              entity(EMPLOYEE)),
    (Works,
              Work-Labour,
              rel(WORKS_IN))},
     University,Employee-Administration)
```



```
construction-expression: {};
reuse-history: {(Ana, 16/03/95, CS-Dept, "OK")};
certification-result: (University-Apps-CI,[<res-p1>,...,<res-pn>]);
reuse-guidelines:
{                                                  /* guidelines for developing a
IF ( Task-Assignment ) THEN                        Task-Assignment application */
                                                   /* integrate the current schema
  REQUIRED:                                        (this) with the one in the
    S=Uφ(this.schema,University-Tasks.schema);     University-Tasks component */
  OPTIONAL:                                        /* retrieve a component
    S1 = CHOOSE IN SELECT schema                   representing Projects concept */
          WITH key-concept = Projects;             /* integrate the schema S with
    S = Uφ (S, S1.schema );                        the one in S1 */
END
                                                   /* guidelines for developing a
IF ( Course-Administration ) THEN                  Course-Administration
  REQUIRED                                         application */
                                                   /* integrate the current schema
  S=Uφ(this.schema,Course-Lecturing.schema);       with the one in Course-Lecturing
END                                                component */
}
```

Figure 2. The `Personal-Administration` reusable schema component.

The `certification-result` states that the applied certification profile (called *certification instance)* is the *University-Apps-CI,* and includes the results for each validated quality property (`<res-pi>`).

The `reuse-guidelines` provide assistance to reuse the component in the development of two specific applications: *Task-Assignment* and *Course-Administration.* When the designer selects the component, he can choose one of the reuse-guideline blocks. These blocks consist of operations processed by the Construction and Selection Tools.

♦

All the objects in the Reuse Environment have been formally specified in an abstract model [Rugg 96]. This model is used to specify the quality properties of the reusable schema components as well as the semantics of the customisation/composition operators. Schema components are implemented in Prolog (including the reuse guidelines). As schema operators are also implemented in Prolog reuse guidelines execution is straightforward.

3. The Conceptual Schema Reuse toolkit.

The Conceptual Schema Reuse (CSR) toolkit implements the above described Reuse Environment, and consists of the following components (Figure 3):

- A *Repository* of certified reusable schema components. The Repository is organised using a Faceted Classification mechanism.
- The *Insertion Tool* that enables the insertion of new reusable components in the Repository
- The *Quality Validation Tool* that enables to certify the quality of the reusable schema components through an enumerative Certification Framework.
- The *Selection Tool* that provides two types of query facilities: (i) a flexible query language based on the Faceted Index and on Domain Models and (ii) a schema-pattern query mechanism.
- The *Construction Tool* that provides a set of operators that enable to build new EER schemas by Customising and Composing the reusable schema components. It also provides mechanisms to assist users, easing the schema construction with reuse tasks.

Schemas developed with reuse are treated in KHEOPS similarly to other schemas developed through other techniques.

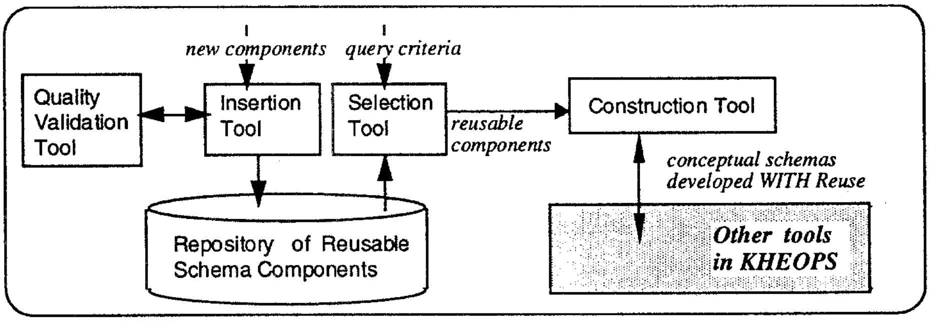


Figure 3. A global view of the CSR Toolkit.

Unlike other proposals of reuse-base development environment, the **Conceptual Schema Reuse (CSR)** toolkit is not an isolated development environment but it is integrated to an existing development environment (KHEOPS). The CSR toolkit not only provides reuse-oriented services to KHEOPS but also re-uses various resources from KHEOPS. More concretely, the CSR toolkit re-uses: the graphic editor to visualise EER schemas [Levr 95], the schema integration primitives [Keda 95], the

Lexical Knowledge Base for Domain Model representation [Ambr 95], the schema comparison primitives and the associated schema similarity editor [Bena 93].

3.1. The Quality Validation Tool.

After the *Insertion Tool* has proceeded to the construction[1] of the reusable components, they must be validated by the Quality Validation Tool. Only after validation will they be inserted into the repository, creating the corresponding links from the Faceted Index.

The quality of reusable components has a great impact on the reuse effectiveness. Components known to be of low or unknown quality will not be reused by developers. On the contrary, high quality reusable software can further help the goals of software reuse by increasing user's confidence in reusable software and by facilitating the application of reuse operations. Moreover, high quality reusable components serve as a basis for developing-with-reuse new good quality software.

The *Quality Validation tool* validates that the reusable components satisfy certain quality conditions required to be member a the Repository. This process is called component certification and it is achieved by using an enumerative Certification Framework. This framework combines a general enumerative certification framework structure [Dunn 92] with specific quality properties of Extended Entity-Relationship schemas. Certification Frameworks consist of a set of *quality properties*, which are predicates on components. Each property has an associated *validation method* which checks if the property holds in a component. The validation methods state how to check the specified property (i.e., the algorithm to validate the property). They can be of different types: *static analysis, formal inspection and formal verification.*

The proposed Certification Framework enables the designer to define different component quality profiles by specifying *Certification Instances*. These play an important role in the Reuse Environment because they serve to state specific qualitative characteristics of the components within the different application domains. In this way, the Certification Framework permits the validation of a set of quality properties in a systematic manner.

A key feature of the framework is flexibility. The idea is not to create a monolithic universal set of validation criteria but rather an extensible framework that can be adapted to different application domains by stating new Certification Instances (i.e., lists of quality properties).

We defined a Certification framework of reusable EER schemas based on the **property classes** stated in Figure 4.

---

[1] In this context, the term *construction* is used as in ADTs theory and Object-Oriented programming: the constructor is a function that builds an object of a type.

Property Classes	**Global meaning (and objective) of the property class**
Syntactic correctness.	- A reusable schema component is syntactically correct if it has the structure stated in the abstract model of KHEOPS.
Syntactic completeness.	- A reusable schema component is syntactically complete if it includes certain characteristics that were not required for syntactic correctness (e.g., cardinalities, reuse-guidelines).
Semantic completeness.	- An EER schema is semantically complete if it represents all the concepts and associations among them in the real world situation.
Consistency.	- An EER schema is consistent if it is possible to assign a no empty instance to each entity and relationship type.
Adequate EER structures	- An EER schema is defined with the adequate EER structures if each real world concept is represented with EER structures according to the semantics of the latter.
Minimality.	- An EER schema is minimal if it uses the minimal set of EER structures to represent adequately the real world situation.
Explicit declarations.	- Certain EER structures require explicit declarations about their semantics. For example, reflexive relationship types and aggregated entity types.
Standardisation	- An EER is standardised if its terms as well as its EER structures representing the real world concepts have a correspondence to a Domain Model.
Complexity.	- An EER schema has a bounded complexity if the values of certain metrics are included in stated intervals. Some of these metrics are: size (the quantity of EER objects), the connection degree of the objects, and the maximum deeps of the specialisation and aggregation hierarchies.
Normalisation	- An EER schema is normalised if it satisfies normal forms as defined in [Bati 92].

Figure 4. The property classes in the proposed Certification Framework.

As stated before, the Certification Framework includes not only properties that can be automatically validated (with static analysis validation method) but also properties that need human interaction to be validated (e.g., properties with formal inspection validation methods).

The Quality Validation tool implements the validation methods for the properties with static analysis ones. This enables an automatic validation of this kind of properties. Concerning the properties that require human interaction, the Quality Validation tool implements a dialogue mechanism which presents the inspector a sequence of steps that he should follow to validate the property and also offers a battery of implemented tests that aids the inspector to evaluate the property and decide if the property holds or not.

3.2. The Selection Tool.

The *Selection Tool* [Ambr 95a] provides three complementary functions: (i) a Semantic Query function that permits the search for reusable components according to the value of their Semantic Descriptor, (ii) a Structure Querying function that permits to recover components whose EER schema is included in a certain schema pattern and (iii) a Browsing function that permits the user to navigate in the Repository. Figure 5 shows the architecture of the Selection Tool.

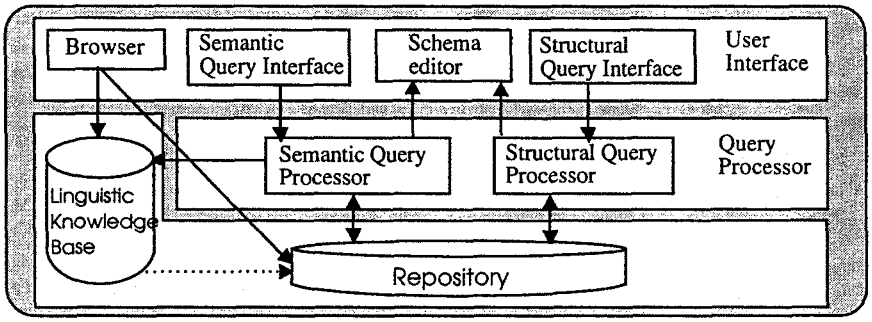


Figure 5. The architecture of the Selection Tool.

The Semantic Query Interface is a graphic interface to the F-SQL query language. In F-SQL, the query conditions concern real world concepts represented in the reusable schema components. *The Structural Query Interface* permits the user to state a skeleton EER schema that defines a (schema) pattern to find in the reusable schemas. This interface is based on the KHEOPS Schema Editor [Dela 95][Levr 95].

The Semantic Query Processor processes the F-SQL queries (for syntax see [Ambr 95a]). It retrieves the reusable components that satisfy the query conditions or that are within a certain *semantic distance* from these conditions. For this, it performs modifications on query conditions.

The Structural Schema Comparison Processor retrieves components with EER schemas included in a certain schema-pattern. For this it computes schema comparisons between the schema-pattern stated by the re-user and the components' EER schema. This comparison is computed using similarity vectors [Comy 92] adapted to treat schema inclusion as to provide structure retrieval.

The Semantic Query Processor performs two types of query modifications: (i) *Relaxations*, which consist in modifications to the Semantic Descriptor values stated in the query condition and which are based on the Linguistic Knowledge Base (LKB) conceptual graph, and (ii) *Reductions*, which consist in removing atomic conditions from the query condition. Table 1 resumes these strategies.

Exact Strategy	• No query modification is applied. • A result is returned only if an exact match is found.

Relaxation Strategy	Deferred Modification	• Query modification is applied one step at a time, based on the modification criteria, until a result is obtained or all permitted modifications are applied, then it stops.
	Forced Modification	• Query modification is applied to the query conditions based on the criteria specified by the user.
Reduction Strategy	(eliminate query conditions)	• Condition reduction is applied one step at a time until a result is obtained, the *mink* is attained, or there is only one condition left, then it stops.

Table 1. The query modification strategies in the Selection Tool.

The traces of the query modifications are re-used in the Construction Tool to approach the schemas to the original query conditions as well as to derive inter-schema correspondence assertions applicable in the merging-composition operations.

3.3. The Construction Tool.

The Construction Tool enables the designer to customise and to compose the selected schemas to build a new one. To achieve this, the Construction Tool supplies the designer with a set of operators, as well as with assistance mechanisms..

The operators supplied by the Construction Tool are classified in two families: the *intra-schema* and the *inter-schema* (or *whole-schema*) operators. The *intra-schema* operators enable to customise the structure of an EER schema by modifying its entity and relationship types. The *inter-schema* operators enable to operate on schemas as a *whole* to achieve schema Union, Subtraction, Intersection and subschema Extraction. Both *intra* and *inter* schema operators return EER schemas as their result.

The *whole-schema* operators enable to achieve schema merging operations. To achieve this, they include the classic Schema Integration (Union operator) as well as other operators that operate on schemas as a whole. Table 2 presents the set of whole-schema operators.

UNION	Union	Performs the integration of two EER schemas.
	Asymmetric Union	A-U_O: The **asymmetric-union-object** of two EER schemas is a new one with the no corresponding structures[2] of each source schemas, and with the corresponding entity and relationship types as in the first schema.

[2] By corresponding structures we mean the ones (attributes, entity and relantionship types) tha represent the same real world concept.

		A-U$_d$: The **asymmetric-union-deep** of two EER schemas is a new one with the corresponding and no corresponding structures of each source schemas, and with the corresponding attributes (in corresponding entity and relationship types) as in the first schema.
SUBTRACTION	Subtraction-o	The **subtraction-object** of two EER schemas is a new one with the entities and relationships in the first schema that are not in the second one.
	Subtraction-d	The **subtraction-deep** of two EER schemas is a new one with the entities and relationships as in the first schema, but without the attributes that are in corresponding structures of the second schema.
INTERSECTION	Intersection-d	The **intersection-deep** of two EER schemas is a new one with the entities and relationships that are in both schemas. Their attributes are also the ones in both source schemas.
	Intersection-u	The **intersection-union** of two EER schemas is a new one with the merge of entities and relationships that are in both schemas.
EXTRACTION	Extraction-object	The **extraction-object** of an entity or relationship type from an EER schema is a new EER schema with this object (entity or relationship type).
	Extraction-hierarchy	The **extraction-hierarchy** creates a new schema with a specialisation or an aggregation hierarchy of a specified entity type in a source schema.
	Extraction-cluster	The **extraction-cluster** creates a new schema with a *cluster* of a specified entity (or relationship) type in a source schema. Two types of *clusters* are applied: a dependency-based and a relationship-grouping.

Table 2. The Whole-schema operators.

Equally to Schema Integration, the *whole-schema* operators need to identify the EER structures in source schemas that correspond to the same real world concepts. We address this issue in two levels: at high-level we define the whole-schema operators as accepting a Boolean function (called ϕ) as a parameter. This function states whether two EER structures in two source schemas correspond to the same concept. At low-level we re-use schema similarity vector techniques for implementing the ϕ function.

The Construction Tool provides two types of assistance mechanisms: (i) automatic schema transformations that approach retrieved schemas to the original query specification, and (ii) the execution of the reuse guidelines stated in the components. The components obtained from a selection can differ from the really searched ones because the Selection Tool transforms the query conditions in order to retrieve potentially reusable components. The differences between the desired and retrieved components may be syntactic (synonyms) or semantic (the retrieved components represent other concepts than the originally searched ones). In order to enhance the understandability of the retrieved components by providing a result closer to the original query conditions, the Construction Tool applies a set of Automatic Transformations that approach the retrieved schemas to the original query specification. These transformations modify the retrieved schemas according to the query modifications performed by the Selection Tool.

Implementation Issues.

The CSR toolkit is implemented in a Unix environment, executing in Sun-Sparc Workstations, and using X11R5/Motif window managers.

All the modules concerning EER schema management (also the Repository) are implemented in Eclipse Prolog, while user-interface parts are implemented in C and C++ using X11/Motif libraries. The tools communicate with the Repository via rpc and pipe mechanisms, and they communicate each other via files. For example, the Selection tool sends the query results to the Construction tool as files with an EER schema each one.

4. Conclusion.

Reuse is recognised as a fundamental technique in Software Engineering. It is also accepted that it is more successfully applied in well-defined application domains which should also have clearly specified models and languages. Other success factors are the facility of obtaining the implementation from the specification, the possibility of reasoning by the use of specifications and the popularity of the specification language. In the Conceptual Modelling domain all these conditions hold. Conceptual Modelling also benefits from the object-oriented paradigm by including its characteristics (e.g. encapsulation, inheritance) in the conceptual models.

Furthermore, in Conceptual Modelling the reusable objects (based on conceptual schemas) are simpler than programs or systems. And some important base techniques needed in the specification of reuse operations have been largely studied in Conceptual Modelling and Databases areas (e.g., schema evolution and integration). These arguments lead us to suppose a successful perspective for a Reuse-Based Conceptual Modelling approach which can be exploited, later on, in a Reuse-based Information System development technology.

In spite of the promising panorama for the application of Reuse in Conceptual Modelling, several technical problems remain to be solved. Although object oriented languages and models provide an important part of the needed capabilities, they do not solve problems related to component certification, classification, selection from a repository, customisation and composition. In addition, Component acquisition remains a hard problem: automated proposals for component acquisition are not developed enough and this activity remains in the hands of domain specialist designers.

This paper deals with some of these topics and proposes a Conceptual Schema Reuse toolkit to apply Reuse techniques in KHEOPS database design environment. The techniques applied in this proposition are inherited from both Software Reuse and Databases areas.

On going research work include the analysis of the interaction between component quality and selection-customisation-composition operations. The objective is to define a notion of *schema component reusability* based on the measurement of the impact that component quality has on operation costs.

5. Bibliography.

[Ambr 95] A.P. Ambrosio, E. Metais , J.N. Meunier, *The Linguistic Level of the Kheops Case Tool,* NLDB'95, Versailles, June 1995.

[Ambr 95a] A.P. Ambrosio *A Semantic Query Mechanism for Conceptual Schema Reuse,* PhD thesis, University of Paris VI, France, June 1995.

[Bati 92] C. Batini, S. Ceri and S. Navathe, *Conceptual Database Design: An ER Approach,* The Benjamin-Cummings Publishing Company, 1992.

[Bena 93] E. Benazet, H. Guehl. *Intégration de vues.* Rapport de Stage DEA, PRISM Laboratory, Versailles, 1993.

[Bigg 89a] Biggerstaff, Richter. *Reusability: Framework, Assessment and Directions,* in [Bigg 89]

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

[Bouz 85] Bouzeghoub M, Gardarin G, Métais E. *Database Design Tools: An Expert System Approach.*, Proc. of the VLDB Conf., Stockholm 1985.

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

[Bouz 94] M. Bouzeghoub, G Gardarin and P. Valduriez, *Du C++ à Merise Objet - OBJETS,* Ed. Eyrolles, 1994.

[Cast 92] Castano, DeAntonelis, Zonta. *Classifying and Reusing Conceptual Schemas.* In Proceedings International Conf. on E/R Approach, 1992.

[Choo 88] Choobineh, Mannino, Numamaker, Konsynski. *An Expert Database Design System Based on Analysis of Forms.* IEEE Transactions on Software Engineering, vol 14, no. 2, 1988.

[Comy 92] I. Comyn-Wattiau and M. Bouzeghoub. *Constraint Confrontation: An Important Step in View Integration.* CAISE'92 Manchester , May 1992.

[Cons 95] Constantopoulos, Jarke, Mylopoulos, Vassiliou. *The Software Information Base: A Server for Reuse.* VLDB Journal, 4, 1-43, 1995.

[Dela 95] E. Delassus, *Conception et Implémentation d'une Interface pour Kheops*, Rapport de Stage DEA, Lab. PRISM, Versailles, 1995.

[Delc 96] Delcambre, Langston. *Reusing (Shrink Wrap) Schemas by Modifying Concept Schemas.* Proc. of Int. Conf. on Data Engineering 1996.

[Dunn 92] Dunn, Knight. *Certification of Reusable Software Parts.* Technical Report, Department of Computer Science, University of Virginia.

[Keda 95] Z. Kedad, *Aspects Linguistiques dans l'Intégration de Vue de Kheops*, Rapport de Stage DEA, PRISM- Université de Versailles, 1995.

[Kers 86] Kersten, Weigand, Dignum, Boom. *A Conceptual Modelling Expret System.* Proc. 5th International Conf. on E/R Approach. Dijon 1986.

[Krue 92] C.W. Krueger, *Software Reuse,* ACM Computing Surveys, Vol. 24, No. 2, June 1992.

[Levr 95] G. Levreau *Un Environnement de Conception, de Documentation et de Maintenance de Schémas de Bases de Données.* PhD thesis, University of Paris VI, 1995.

[Li 96] Wen-Syan Li, Richard D. Holowczak. *Constructing Information Systems Based on Schema Reuse.* Procs. of the CIKM'96 Conf. 1996.

[Louc 92] P. Loucopoulos and R. Zicari eds., *Conceptual Modelling, Databases, and Case - An Integrated View of Information Systems Development,* John Wiley and Sons, Inc., 1992.

[Louc 92a] P. Loucopoulos. *Conceptual Modelling.* in [Louc 92].

[Meta 93] Métais, Meunier, 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.

[Mili 95] Mili, Mili, Mili. *Reusing Software: Issues and Research Directions.* IEEE Transactions on Software Engineering, 21(6), June 1995.

[Prie 87] R. Prieto-Diaz, *Classification of Reusable Modules,* in [Bigg 89].

[Prie 91] Prieto-Diaz, Arango. *Domain Analysis and Software System Modelling.* IEEE Computer Society Press. Los Alamos - California, 1991.

[Roll 92] C. Rolland and C. Cauvet, *Trends and Perspectives in Conceptual Modelling,* in [Louc 92].

[Rugg 96] Ruggia. *Applying Reuse in Conceptual Modeling: a toolkit approach.* Phd thesis, University Paris 6 - France. April 1996.

[Seo 94] Seo, Loucopoulos. *Formalisation of Data and Process Model Reuse Using Hierarchic Data Types.* Proc. of CAiSE 1994.

[Thun 92] Thunheim, *Development with and for Reuse: Guidelines from the REBOOT Project,* In ERCIM Workshop Report: Methods and Tools for Software Reuse, Heraklion, October 1992.