Design of Meta Database Model and Universal Business Applications

A. Murdjeva, M. Tsaneva

Key Words: *Data base design; business logic; abstract design; universal application.*

Abstract. *Nowadays the activities automation in a specific business area is not technological news, but a business need. The necessity of more and more business applications implies the quest for tools and techniques for fast and efficient development of information systems. Rapidity and efficiency are two important directions, in which business application development technologies are developing today, aiming at these two important qualities being achieved regarding both development process stages and system exploitation. The effective exploitation is related to real usage of the information system. And this is available when information decision follows the pace, in which processes in the business area develop. The achievement of this aim is a new challenge to the technologies. In this paper, some ideas are presented about changing the patterns of database design, directed to produce applications which are maximum adaptive and scalable.*

1. Introduction

Problems during Business Applications Development

The fact that the world of business today is binding more and more tightly with the automation of information processing is obvious. This commitment is also revealing in the influence of the business over applications, by implying its limitations and requirements, and changing business processes as a result of their automation. This tight commitment originates some problems, which in general terms may be defined as: *Business is changing and is abounding in new processes, but it is not easy for business applications to follow this natural progress.*

A much more detailed glance on the problems, which business applications developere are facing is necessary. This would allow to discover directions in which the cross-purposes between business and applications are most significant and to find out a new way of their development.

Three main problems which have both their own abstract and philosophical explanation and concrete practical behavior may be defined.

• The problem of indefiniteness and narrowness of the expert knowledge [4]

During the process of creation the information system developers are meeting with insufficient, relatively imprecise and unstructured knowhow about the business area.

One possible solution of this problem is the development of generalized data structures, which can describe (at some stage) a highly varied expert knowledge and which allow relatively simple addition and adjustment, preserving their correctness.

• The problem of changing the point of view

This problem has two sides, which may be defined as the *problem of different points of view and the problem of changing a particular point of view.*

This problem is directly related to the dynamics of the business environment, some activities of which an application is intended to automate. If we examine the user needs and the ideas of automation, which these needs originate, as dynamic entities with their own life cycle in practice this life cycle is very short because of the rapid change of user needs.

This enforces to find out some tools, technologies and approaches for information systems development, which provide that the shortened life cycle of users need does not significantly change the life cycle of the information system.

• The problem of similarity

During the elaboration of an information system, the similarities between current and preceding design and implementation of individual items are often unnoticed. Their un-discovering leads to over-again creation of uniform already existing items, which costs recourse and time loses.

An alternative for overcoming this same problem is finding out a general approach for examination of business area entities.

This article suggests an approach to data base design for business applications, as well as a mechanism for business logic control in these applications. The aim is to suggest a tool and a technique to develop flexible and independent applications, isolated from particular business problem and usable in different areas.

2. Meta Model

The relational data base does not permit to change the data structures in a business application easy enough, without causing significant corrections in the application itself. On the other hand, the relational data base is enough abstract regarding these structures from the point of view of their maintenance, i.e. it guarantees equally good enough the referential integrity through foreign keys, uniqueness through primary keys, correct result of queries and so on, without being interested in the particular problem, which has been described by the tables in the user's scheme. Data Base Management System (DBMS) achieves this by virtue of meta description of its own structures [8].

Similar approach is necessary for data description, so that the application can transform itself into an abstract interpreter of these data.

Applying meta-design during data structures design, through which to abstract objects and their descriptions is the sugges-

tion made in this paper. **With meta-design the data base converts into a united structure for storing business objects Meta data and their particular instances.**

A fundamental meta-design approach is business objects abstraction and treatment not from the point of view of their semantics in the business process, but regarding their more general role in it. This more general role is to describe entities. So the point of view to the object description is changing. Their description is not specific, but is a meta-description – *description of their description.*

Proceeding from this assumption and from the necessity to preserve main functionality of the DBMS, especially the ability to distinguish object description and object identification, the following structures for meta-design are necessary and sufficient:

✓**Types.** A nomenclature of different business objects, which are described in the data base. It is expandable and is needed to provide object diversion and typification of their description.

✓**Entities.** A nomenclature of all business objects, described in the data base. It is arbitrarily expandable and does not contain a concrete description of these objects.

✓**Properties.** A nomenclature of necessary properties of all business objects, which are interesting for the business application. This nomenclature is also arbitrarily expandable and thanks to this, the description of business objects can easily be added with new object properties. To guarantee the quality and correct interpretation of the data, each attribute must be defined regarding:

- Obligatory (Mandatory) for object description.
- Repeatability within one and the same object description (Single or Multiple).
- Relationship with other objects (i.e. description of relationships between objects in the 1 to M direction). The relation of type child-parent is presented.
- Value Type – the type of the value, with which it instantiates in the description.
- Additional constraints on the value (Checks).

✓**Templates.** To ensure an adequate attribute set, but not a general description, through which to provide the possibility of specific interpretation of attributes of the same name belonging to different entity types, the meta model maintains a template of each business object. The template is a concrete attribute set for a specific entity type. These templates are expandable in the terms of a single type. They can be supplemented with new attributes or the existing attribute set may be revised. In conjunction, the template nomenclature is also expandable regarding supplementation with new templates for new object types.

✓**Description.** Through this structure the storage of each object concrete attribute values is ensured. The content of this

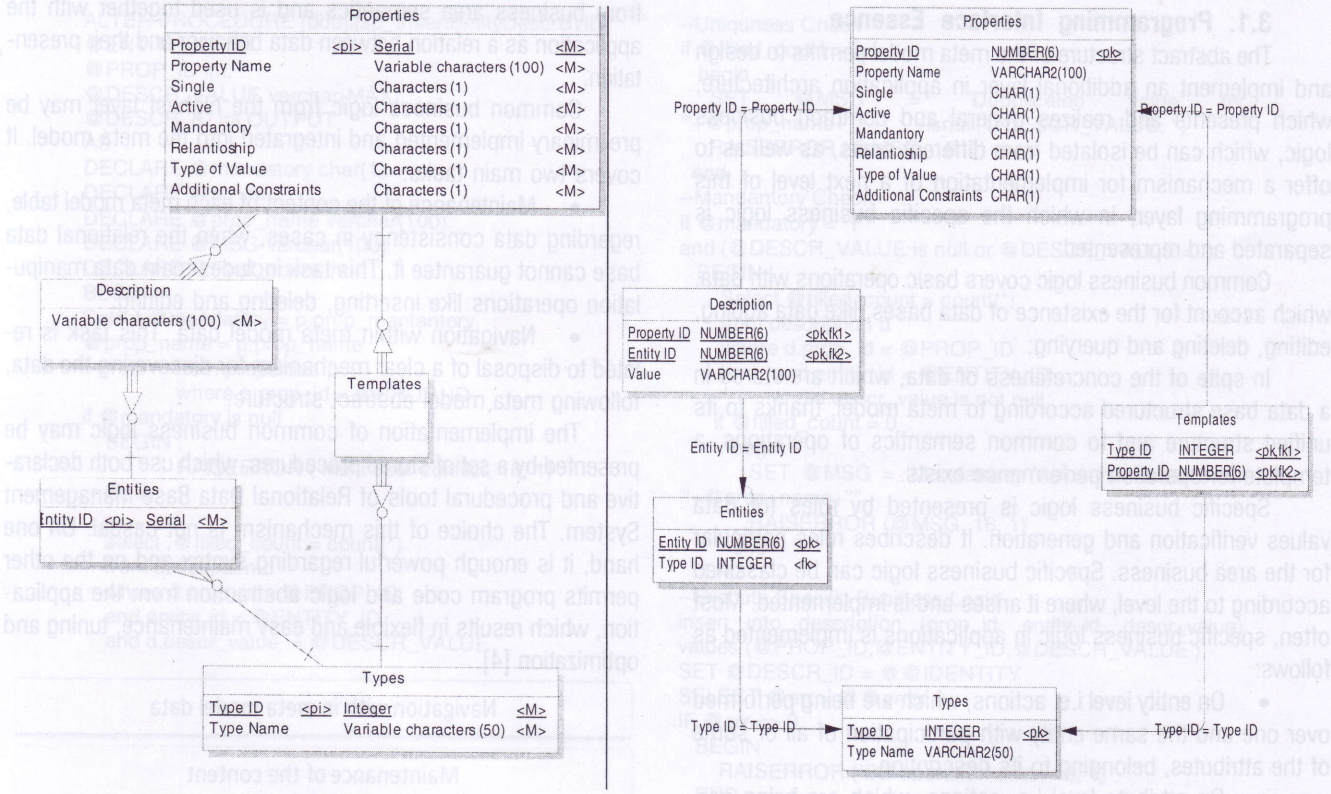

Figure 1. Logical and physical data base meta model

structure is controlled by the attribute properties, which set in their nomenclature.

A data model, created applying the approach of meta-design looks as shown on *figure 1*. It does not concretize a particular business area, but using it many different areas may be described.

The application of the meta model changes the tasks, which are delegated to the application. Under classic design, the application has as a main task to present the data in conjunction with the implementation of a significant part of business logic related to data value validation and other processing. Current three-layer application architecture permits the application to discharge from business logic, as it is delegated to a new layer in the architecture. But nevertheless, this does not provide the possibility of independent and flexible application development, so applications are always tightly coupled with business area and maintenance process. Meta model permits suggestion of analysis and design techniques and tools to produce applications, which are independent from business specifics, to produce application shells, in which business specific functionality is achieved by data entry and maintenance. This way, the application is responsible just for presentation and meta model is the mediator between it and business logic.

3. Programming Interface for Meta Data Maintenance and Usage

3.1. Programming Interface Essence

The abstract structure of the meta model permits to design and implement an additional layer in application architecture, which presents and realizes general and common business logic, which can be isolated from different areas, as well as to offer a mechanism for implementation of a next level of this programming layer, in which the specific business logic is separated and represented.

Common business logic covers basic operations with data, which account for the existence of data bases, like data adding, editing, deleting and querying:

In spite of the concreteness of data, which are stored in a data base structured according to meta model, thanks to its unified structure and to common semantics of operations, a template for operation performance exists.

Specific business logic is presented by rules for data values verification and generation. It describes rules particular for the area business. Specific business logic can be classified according to the level, where it arises and is implemented. Most often, specific business logic in applications is implemented as follows:

- On entity level i.e. actions, which are being performed over one and the same entity with participation of all or some of the attributes, belonging to its description.
- On attribute level i.e. actions, which are being performed over a concrete value, belonging to the description of a business area entity, aiming its verification or calculation.

Via meta model usage in data base design for business applications, the programming of application business logic is reduced to data tuning, i.e. to configuration of main tables contents, providing the data base with knowledge of the relations between data and their specific business logic. This relation defines which program code must be executed during value adding or editing of each attribute. The essence of this specific business logic is an abstract algorithm, which can be completely defined and implemented for a vast range of business areas.

3.2.Programming Interface Implementation

The implementation of common and specific business logic itself may be done by a new programming interface to the meta model. This programming interface has its own architecture, which is determined by the process of abstracting the specific from the common business logic and consists of several layers. Each higher layer has higher degree of generalization and expresses more the common business logic, then the specific one.

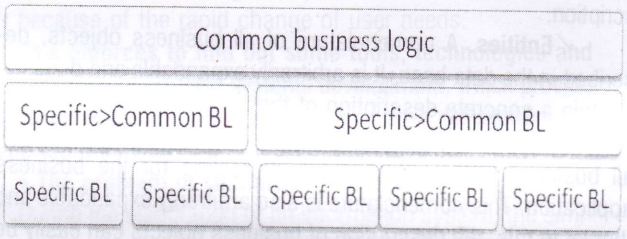

Figure 2. Business logic architecture

The highest layer of this architecture is mostly isolated from business area semantics and is used together with the application as a relation between data behavior and their presentation.

Common business logic from the highest layer may be preliminary implemented and integrated into the meta model. It covers two main tasks:

- Maintenance of the content of each meta model table, regarding data consistency in cases, when the relational data base cannot guarantee it. This task includes main data manipulation operations like inserting, deleting and editing.
- Navigation within meta model data. This task is related to disposal of a clear mechanism for discovering the data, following meta model abstract structure.

The implementation of common business logic may be presented by a set of stored procedures, which use both declarative and procedural tools of Relational Data Base Management System. The choice of this mechanism is not casual. On one hand, it is enough powerful regarding syntax and on the other permits program code and logic abstraction from the application, which results in flexible and easy maintenance, tuning and optimization [4].

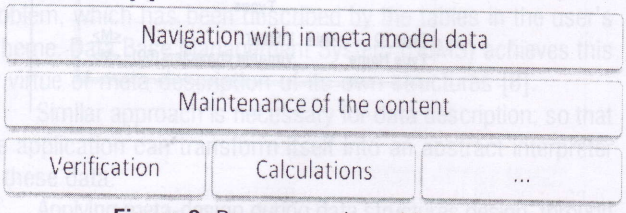

Figure 3. Program code architecture for meta model usage

Maintenance of the content includes basic abstract operations for data adding, editing and deleting in each meta model entity, which implement the following functions:

- To guarantee the mandatory of dynamic values.
- To guarantee the uniqueness of all entered values - both of dynamic attributes.
- Control over dynamic value entry, according to specific business rules, implemented as a part of specific business logic.
- Control over data consistency.

Each of these common algorithms may communicate (call in a standard way) with preliminary implemented specific business logic.

An example for procedures implementing common business logic for content maintenance is presented on *figures 4* and *5*.

The operation of deleting meta model data performs the main task related to maintenance of dynamic data consistency and its implementation is presented on *figure 6*.

Operations related with navigation within meta model data perform the main task of discovering entities by their description. The navigation within meta model is an operation of very high level of abstraction.

This is achieved by examining the searching criteria set and finding them out in the model as a specific type of entity, which specific type of entity has also an analogical 'attribute-value' description i.e. uses the meta model itself for searching criteria definition. Creation of such an entity is done using the programming interface, which implements the common business logic - adding new searching criteria set (new entity) and adding new lookup values (entity description). The algorithm for searching is based on the data of one entity of this type and transforms the logical relations between criteria (AND, OR) into plural operations section and union.

The lookup operation may be reduced to meta model data filling and has two forms:

- Searching for a particular entity by its identification.
- Searching for a set of entities by criteria set.

Searching for a particular entity has as a main task to find out the attribute set, which describes an entity including both attributes with values set and empty attributes. Meanwhile it performs the specific task to retrieve the lookup criteria (according to abstract interpretation of this search). An example for procedures, which implement common business logic for searching a concrete entity, is presented on *figure 7*.

Looking up for a set of entities, which satisfy criteria set by applying plural operations comprises of:

- Creating an entity of specific type (f.e. Filter).
- Filling in its description, which serves as a lookup criteria set.
- Performing a general algorithm to form a dynamic data base query according to the criteria being specified.

An example of procedures, implementing common business logic for searching a particular entity is presented on *figure 8*.

```
ALTER PROCEDURE [dbo].[sp_InsertEntityDescrByID]
@ENTITY_ID int,
@PROP_ID int,
@DESCR_VALUE varchar(MAX),
@DESCR_ID int OUTPUT
AS
DECLARE @mandatory char(1)
DECLARE @err int
DECLARE @prop_name varchar(100)
DECLARE @MSG varchar(100)
DECLARE @filled_count int
BEGIN
Select @mandatory = p.prop_mantantory,
@prop_name = p.prop_name
        from properties p
        where p.prop_id = @PROP_ID
if @mandatory is null
  BEGIN
        RAISERROR ('Multi user conflict, 16, 1)
  END

  select  @filled_count = count(*)
  from description d
  where d.prop_id = @PROP_ID
  and entity_id = @ENTITY_ID
  and d.descr_value  = @DESCR_VALUE
```

```
--Uniquness Check
if @filled_count > 0
  begin
    SET     @MSG     =     "Dupplicated     value     for
'"+@prop_name+"" + ' : '+isnull(@DESCR_VALUE, '')
    RAISERROR (@MSG, 16, 1)
  end
--Mandantory Check
if @mandatory = 'Y'
and (@DESCR_VALUE is null or @DESCR_VALUE = '')
  BEGIN
     Select @filled_count = count(*)
     from description d
     where d.prop_id = @PROP_ID
          and entity_id = @ENTITY_ID
          and d.descr_value is not null
    if @filled_count = 0
      begin
        SET @MSG = 'Mandantory Value not eneterd for
'"+@prop_name+""
        RAISERROR (@MSG, 16, 1)
      end
  END
--Execute Specific Business Logic
insert into description (prop_id, entity_id, descr_value)
values (@PROP_ID,@ENTITY_ID,@DESCR_VALUE )
SET @DESCR_ID = @@IDENTITY
SELECT @err = @@error
IF @err <> 0
  BEGIN
     RAISERROR ('DB operation error, 16, 1)
  END

END
```

Figure 4. Program code for data adding, implemented by a stored procedure

```
ALTER PROCEDURE [dbo].[sp_UpdateEntityDescrByID]
@DESCR_ID int,
@DESCR_VALUE varchar(MAX)
AS
DECLARE @err int
DECLARE @mandatory char(1)
DECLARE @prop_name varchar(100)
DECLARE @MSG varchar(100)
DECLARE @filled_count int
BEGIN
Select @mandatory = p.prop_mantantory,
@prop_name = p.prop_name
from properties p, description d
where d.descr_id = @DESCR_ID
and p.prop_id = d.prop_id
if @mandatory is null
  BEGIN
    RAISERROR ('Multi user conflict, 16, 1)
  END
        select  @filled_count = count(*)
        from description d
        where d.prop_id =
                (select prop_id
                from description
                where descr_id = @DESCR_ID)
        and entity_id =
                (select entity_id
                from description
                where descr_id = @DESCR_ID)
        and d.descr_value  = @DESCR_VALUE
        and d.descr_id <> @DESCR_ID
if @filled_count > 0
   begin
     SET    @MSG    =    'Dupplicated    value    for
'"+@prop_name+"' + ': '+isnull(@DESCR_VALUE, '')
     RAISERROR (@MSG, 16, 1)
end

if @mandatory = 'Y'
and (@DESCR_VALUE is null or @DESCR_VALUE = '')

BEGIN

  Select @filled_count = count(*)
  from description d
  where d.prop_id =
          (select prop_id
             from description
             where descr_id = @DESCR_ID)
          and entity_id =
          (select entity_id
             from description
             where descr_id = @DESCR_ID)
          and d.descr_value is not null
  if @filled_count = 0
    SET  @MSG  =  'Mandantory  Value  not  eneterd  for
'"+@prop_name+"'
    RAISERROR (@MSG, 16, 1)
 END
          update description
          set descr_value = @DESCR_VALUE
          where descr_id = @DESCR_ID

SELECT @err = @@error
IF @err <> 0
  BEGIN
    RAISERROR ('DB operation error ', 16, 1)
 END


END
```

Figure 5. Exemplary program code for data editing, implemented by a stored procedure

```
ALTER PROCEDURE [dbo].[sp_DeleteEntityByID]
@ENTITY_ID int
AS
DECLARE @err int
DECLARE @E_TYPE int
DECLARE @USED_COUNT int
DECLARE @MSG varchar(100)
BEGIN
Select @E_TYPE = e.type_id
from entity
where entity_id = @ENTITY_ID

Select @USED_COUNT =
          count(*)
          from description d, properties p
          where d.descr_value = @ENTITY_ID
          and prop_id  = prop_id
          and prop_rel_type = @E_TYPE

if @USED_COUNT > 0
   BEGIN
     SET @MSG = 'Entity used'
     RAISERROR (@MSG, 16, 1)
   END

delete from  entity
where entity_id = @ENTITY_ID

SELECT @err = @@error
   IF @err <> 0
   BEGIN
     RAISERROR ('DB operation error', 16, 1)
   END
END
```

```
ALTER PROCEDURE [dbo].[sp_DeleteEntityDescrByID]
        @DESCR_ID int
AS
DECLARE @err int
BEGIN
delete from  description where descr_id = @DESCR_ID
SELECT @err = @@error
  IF @err <> 0
    BEGIN
      RAISERROR (DB operation error', 16, 1)
    END
END
```

Figure 6. Exemplary program code for deleting, implemented by a stored procedure

```
ALTER PROCEDURE [dbo].[sp_GetEntityDescrById]
@ENTITY_ID int
AS
BEGIN
if @ENTITY_ID < 0
        if @ENTITY_ID = -100
          Select p.prop_id as prop_id, null as entity_id,
          '' as descr_value, null as descr_id,
          p.*, @ENTITY_ID as entity_type,
          '' as  descr_text,
          isnull(p.prop_order, 99) as prop_new_order
          from properties p
          where p.prop_active = 'Y'
          order by 17 asc, 3 asc
        else
          Select p.prop_id as prop_id, null as entity_id,
          '' as descr_value, null as descr_id,
          p.*, t.type_id as entity_type,
          '' as  descr_text, isnull(p.prop_order, 99)
          from template t, properties p
          where t.prop_id= p.prop_id
          and t.type_id = @ENTITY_ID*(-1)
         order by 17 asc, 3 asc

else

Select d.*,p.*, e.type_id as entity_type,
isnull((select dbo.uf_entity_descr(er.entity_id)
          from entity er
          where er.entity_id = cast(d.descr_value as int)
          and p.prop_relantionship = 'Y'), d.descr_value) as
          descr_text,
isnull(p.prop_order, 99) as prop_new_order
from description d, properties p, entity e
where d.prop_id= p.prop_id
and e.entity_id = d.entity_id
and d.entity_id = @ENTITY_ID
union all
Select p.prop_id, @ENTITY_ID, '', null, p.*, e.type_id as
entity_type, null, 99
from properties p, entity e, template t
where p.prop_id not in
          (select prop_id from description d
          where     e.entity_id = d.entity_id)
          and e.entity_id = @ENTITY_ID
          and t.type_id = e.type_id
          and t.prop_id = p.prop_id
          order by 17 asc, 3 asc
END
```

Figure 7. Program code for searching a concrete entity implemented by a stored procedure

```
CREATE PROCEDURE [dbo].[sp_GetListByFilter]
@FILTER_ID int
AS
DECLARE @cursor CURSOR
DECLARE @sql varchar(max)
DECLARE @prop_id int
DECLARE @descr_value varchar(max)
DECLARE @count int
DECLARE @sql_oper varchar(max)
BEGIN
IF @FILTER_ID is not null
BEGIN
  SET @cursor = CURSOR FOR
  select prop_id, descr_value
  from description
  where entity_id = @FILTER_ID
SET @sql =''
SET @count = 1
SET @sql_oper = ''
OPEN @cursor
FETCH NEXT FROM @cursor
INTO @prop_id, @descr_value
WHILE @@FETCH_STATUS = 0
BEGIN
 IF @count > 1
SET @sql_oper = 'intersect'
SELECT @sql =
case @prop_id
    when -1 then
        @sql + @sql_oper + ' select entity_id from
description d where d.entity_id = '+ @descr_value
    when -2 then
        @sql + @sql_oper + ' select entity_id from entity d
where d.type_id = ' + @descr_value
    Else


@sql + @sql_oper+ ' select entity_id from description d
where d.prop_id = ' + cast (@prop_id as varchar)+ ' and
d.descr_value like ''% ' + @descr_value + '%'''
  end
  FETCH NEXT FROM @cursor
  INTO @prop_id, @descr_value
  SET @count = @count + 1
  END
  SET @sql = @sql + 'select e.entity_id,
dbo.uf_entity_descr(e.entity_id), et.type_name from entity e
, entity_types et where e.type_id = et.type_id and entity_id
in ('+@sql+')'

 EXEC (@sql)
 END
ELSE
 BEGIN
  select e.entity_id, dbo.uf_entity_descr(e.entity_id),
  et.type_name
  from entity e, entity_types et
  where e.type_id = et.type_id
  union all
  select '', 'empty' , ''
  order by 1 desc
END

DEALLOCATE @cursor

END
```

Figure 8. Program code for searching entity set, implemented by a stored procedure

A generalization of program code architecture, implemented by stored procedure is illustrated on *figure 9*.

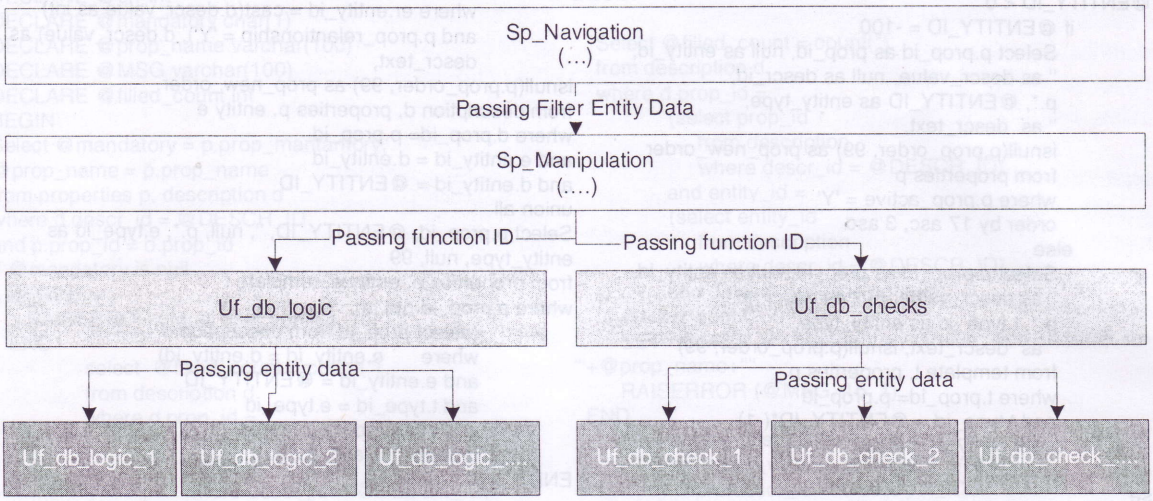

Figure 9. Program code architecture for meta model data maintenance

4. User Interface for Meta Data Usage and Maintenance

Taking the business logic out of application, appropriating it to the description attributes and implementing it by a specific data base function, permits the design and development of an application, which is as much as possible independent of area and its business processes changes.

The high degree of meta model abstraction allows the investigation of decisions directed to design and implementation of a user interface with unified functionality, applicable to a wide range of business areas. This means to suggest a universal user interface, which contains the necessary and sufficient functionality, according to common business logic, which is charged with the execution of abstract operations on data, and which is not 'interested' in a particular business problem. The abstract approach suggested, allows an easy implementation of a user interface with unified preliminary defined functionality and variable set of user forms for visualization, which are placed over abstract functionality. While using meta model, the process of user interface design and implementation is transformed to a process of user interface tuning, which tuning is basically related with data entry, and requires less programming.

An application based on the meta model suggested, may provide the needed functionality by a minimal data entry forms set of two basic types:

- A form for viewing/editing/filtering of a single business entity *(figure 10)*.

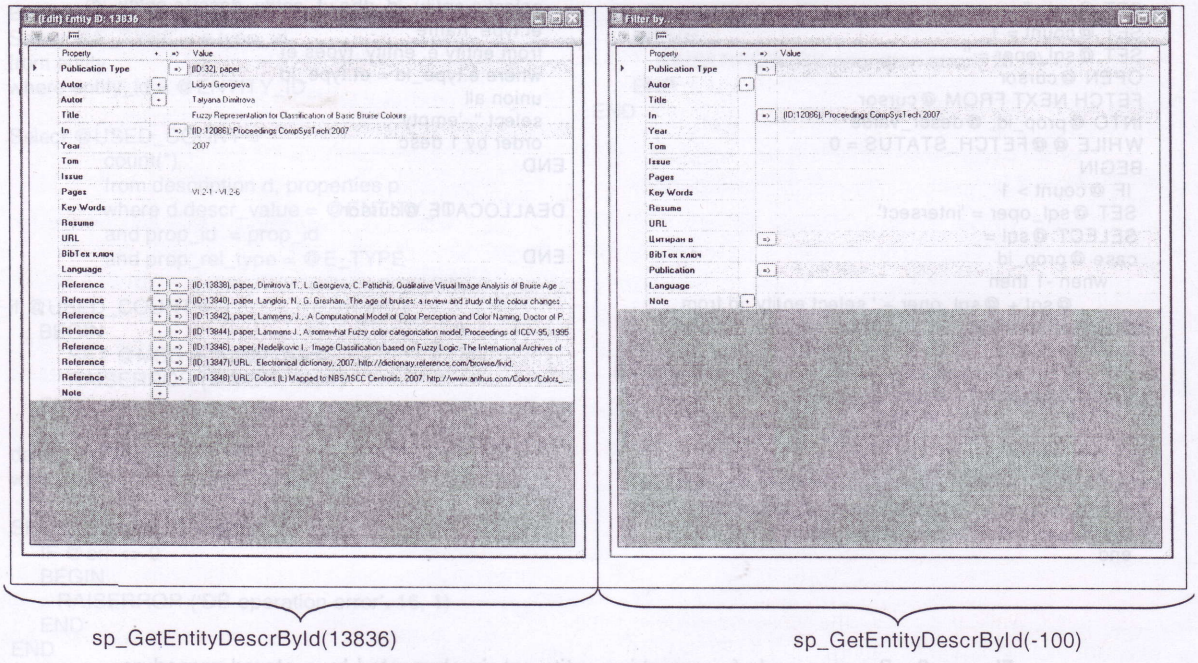

Figure 10. Form business entity maintenance

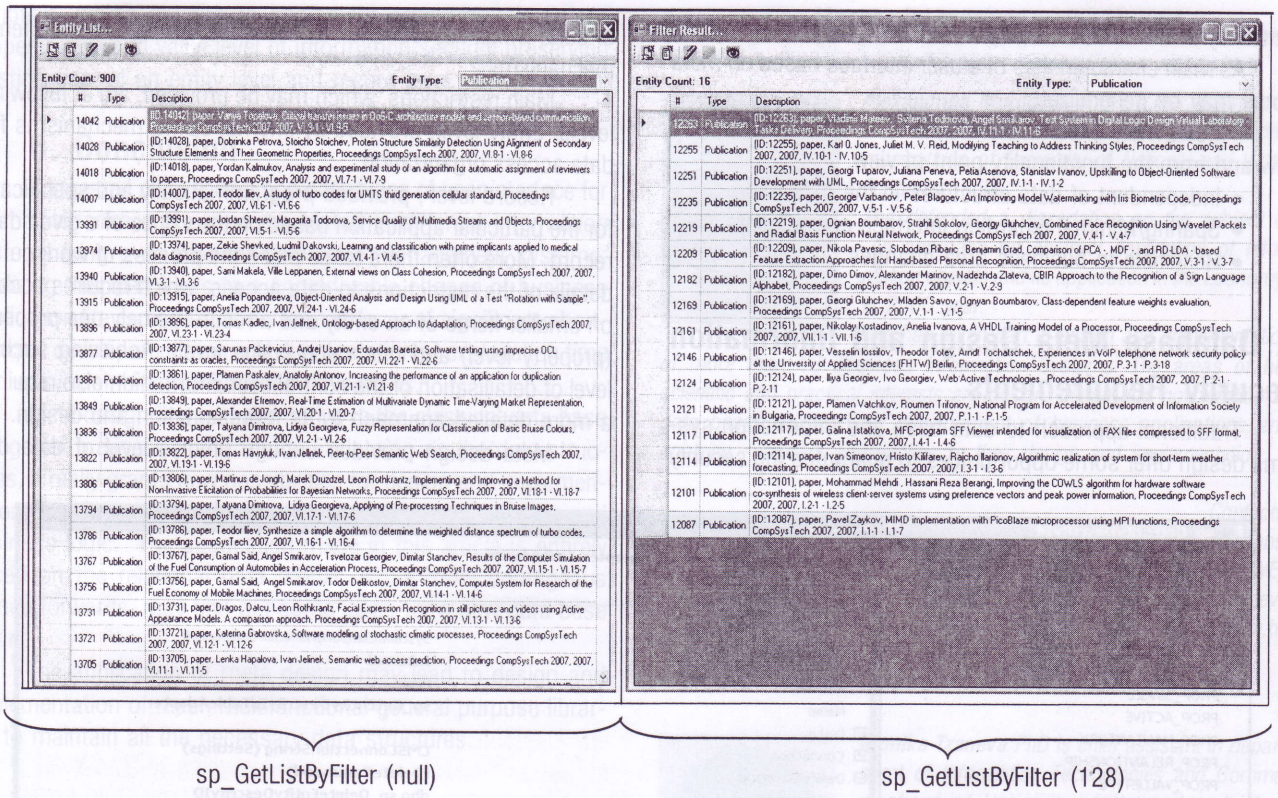

Figure 11. Form List of business entities

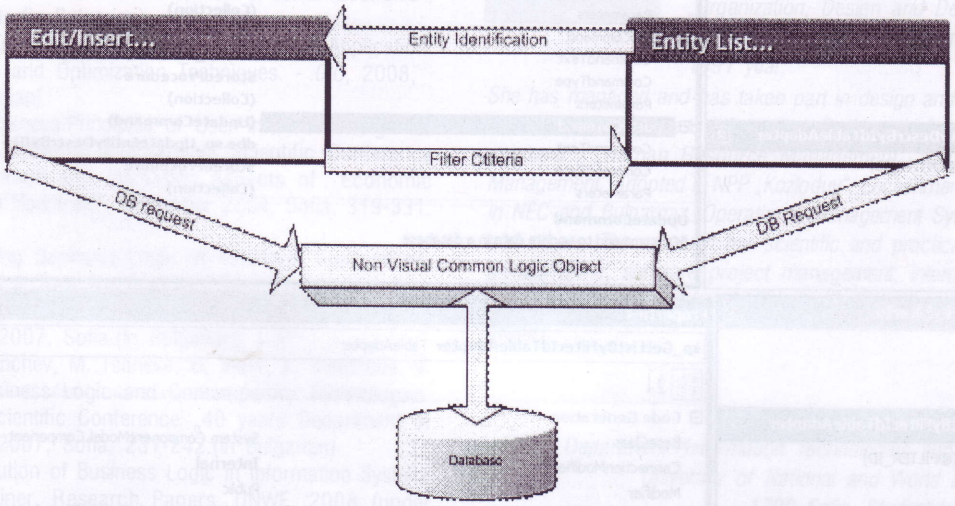

Figure 12. Interaction between user interface forms in the business application

• A list of business entities, obtained after applying a filter (all entities or entities satisfied the criteria set) as shown on *figure 11*.

Both types of forms are bound by a mechanism for parameter delivery aiming to identify the entity type - entity of type Filter or entity for data manipulation.

Implementing the possibility to use different presentation styles over one and the same data structure and functionality becomes possible by integrating of a non visual component in the user interface architecture. It implements the binding with the highest abstract layer of programming interface and it delivers data to the forms for presentation and respectively delivers data to the meta model. This layer in the architecture ensures the next level of isolation of the application from the business logic and the isolation of presentation from processing *(figure 12)*.

An exemplary implementation of a non visual object, which provides the basic functionality over meta model may be pre-

sented by a .NET DataSet component.

As main characteristics of a user interface based on meta model may be generalized:

- Completely expandable - both from the data point of view and from the functionality point of view.
- Independent of data structures in the data base
- Sparing/Minimal regarding interface components
- Resistible to business area changes

5. Database Meta Design and Information Security Requirements

Traditional approach to data base and information systems design offer some opportunities to guarantee the security of data, which are restricted by database management systems mechanisms.

Main restrictions, which may be provided, are at following levels: Entity level, provided by internal DBMS mechanisms for data access organization.

Record level , provided programmatically and specifically for the particular application based on the values of a given data record. More often there takes place the process of work out in details of the restrictions to data access, which requires accent on smaller items of an entity and a record namely one property (property level) and one value (value level). Achieving such a level of detailisation of the control over information also requires a more detailed approach to its representation and design.

Meta design provides enough precise level of data de-

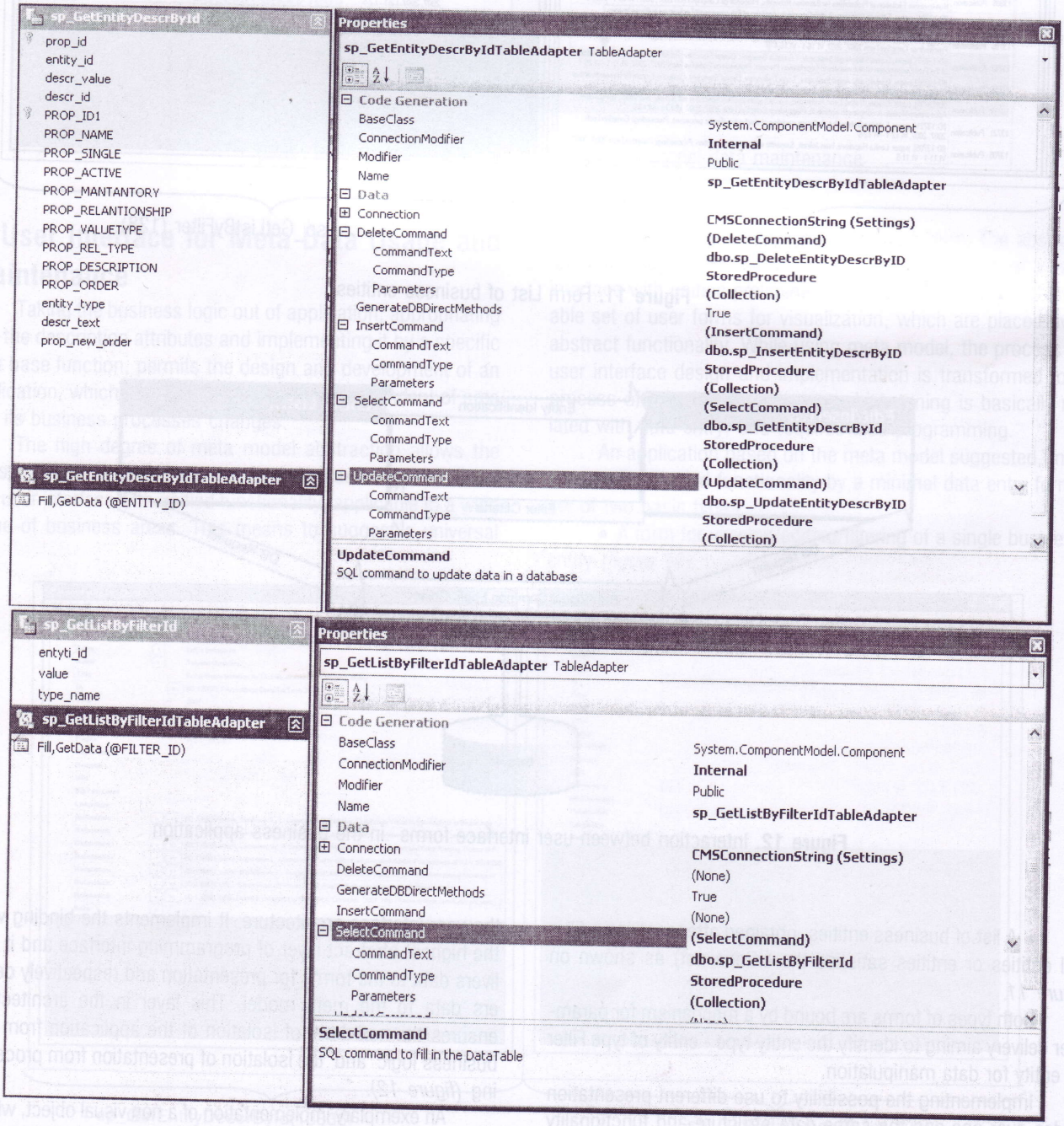

Figure 13. Non visual common logic object

scription, which permits to implement programmatically the property level and value level of information security preserving existing control on entity level and record level.

6. Conclusion

Suggested meta model and both types of user interface for its maintenance and usage are tested in different information systems lying over different data base management systems. Realizing this idea leads to a considerable change of business application life cycle stages duration in favor of significant shortening of development time and augmentation of exploitation time with easy maintenance from the developers.

Meta design is a real alternative to the classical data base design for business applications. Endorsing on the data structures, it really grants new possibilities for design and implementation of all items of application architecture - user interface and business logic. The suggestion made in this article is entirely based on the relational data model, but a more courageous glance forwards, may find out some ideas for a new data base engine.

The application of meta design may lead to design and implementation of reach multifunctional general purpose libraries to maintain all the necessary data structures.

References

1. Jeliazkov, J., A. Murdjeva. Unified Format for Storing Data and Data Integration. International Scientific Conference „Management, Information and Marketing Aspects of Economic Development of Balkan Countries", Sofia, November 2004. (in Bulgarian)
2. Murdjeva, A., M. Tsaneva. Optimization of Business Application. Aspect of Optimization and Optimization Techniques. - *CIO*, 2008, No 4, 90-93.(in Bulgarian)
3. Murdjeva, A., M. Tsaneva.Principles of User Interface Design for Business Information Systems. International Scientific Conference „Management, Information and Marketing Aspects of Economic Development of Balkan Countries", November 2004, Sofia, 319-331. (in Bulgarian)
4. Murdjeva, A. Splitting Business Logic in Multilayer Applications. Stored Procedures as a Tool for Business Logic Implementation. Preprints of Jubilee Scientific Conference „40 years Department of Informatics", October 2007, Sofia.(in Bulgarian)
5. Murdjeva, A., E. Denchev, M. Tsaneva, D. Velev, K. Stefanova, V. Lazarova. Splitting Business Logic and Contemporary Technologies. Preprints of Jubilee Scientific Conference „40 years Department of Informatics", October 2007, Sofia, 231-242.(in Bulgarian)
6. Murdjeva, A. Distribution of Business Logic in Information System. Business Logic Container. Research Papers, UNWE, 2008 (under print).(in Bulgarian)
7. Murdjeva, A. Symilarity and Dynamics of Information Objects. Design of Unified Relational Structures. Research Papers, UNWE, 1999.(in Bulgarian)
8. Tsaneva, M., A. Murdjeva. User Useful Report – Strategies for Design. Symposium Articles „35 Years Specialty of Informatics", UMWE, Sofia, 2003, 175-183.(in Bulgarian)
9. Tsaneva, M., A. Murdjeva. Technology for Invoice Generation Based on Dynamic Data about Clients Duties Balance.-*Bulgarian Accounter*, 2008, No. 6.(in Bulgarian)
10.Tsaneva, M. Organisation of User Oriented Operations Log in Business Information Systems.- *CIO*, 2008, No. 3, 75-78.(in Bulgarian)
11.Gennick, J. The Master Key to Oracle's Data Dictionary. http:/www.oreillynet.com/pub/a/network/2002/10/28 data_dictionary.html, 2002.

Manuscript received on 07.07.2008 r.

Alexandrina Murdgeva *PhD is chief assistant in department of Information Technologies and Communications of University of National and World Economy, Sofia.*

She defended a dissertation on the subject of „An approach for abstract description of information objects and its application in the Economy" in 2000 year.

She has taken part in design and implementation of more than 10 Business Information Systems. The areas of her scientific and practical interests are design, usage and optimization of data bases, design and development of business applications and applying abstract approaches to their design and implementation.

Contacts:
Department "Information Technologies and Communications"
University of National and World Economy (UNWE),
1700 Sofia, Studentski grad "Hr. Botev",
e-mail: amurjeva@abv.bg

Monika Tzaneva *PhD is chief assistant in department of Information Technologies and Communications of University of National and World Economy, Sofia - she teaches System programming, Software Project Management and Informatics.*

She defended a dissertation on the subject of „Organization, Design and Development of Object-oriented Distributed Systems in Economy" in 1991 year.

She has managed and has taken part in design and implementation of more than 15 Business Information Systems incl. „Nuclear Fuel Management", „Human Resource Management" and „Document Flow Management" adopted in NPP „Kozloduy", Encashment systems adopted in NEC and Bulgargaz, Operational Management System of TBI Credit and others. The areas of her scientific and practical interests system programming, software project management, interoperability of business applications, SOA, CASE technologies for system analysis and design.

Contacts:
Department "Information Technologies and Communications"
University of National and World Economy (UNWE),
1700 Sofia, Studentski grad "Hr. Botev",
e-mail: monika_tzaneva@yahoo.com