Exploring the Essence of an Object-Relational Impedance Mismatch

- A novel technique based on Equivalence in the context of a Framework

Christopher Ireland, David Bowers, Michael Newton, Kevin Waugh
Department of Maths and Computing
The Open University
Milton Keynes, UK
{cji26@student.open.ac.uk, D.S.Bowers@open.ac.uk, M.A.Newton@open.ac.uk, K.G.Waugh@open.ac.uk }

***Abstract*- During the development of an object-relational application we combine technologies that make use of object and relational artefacts because each is suited to a particular role. However such a combination of technologies gives rise to problems of an object-relational impedance mismatch. In this paper we highlight these problems arise not just because of differences in language or design objective, but because the semantics and data of an object and a relational artefact are not equivalent. We introduce a novel technique based on equivalence, and use this to explore one problem of an object-relational impedance mismatch. We show that strategies for dealing with the problem of identity should not focus on a correspondence between the two identity systems but on a correspondence between the different ways in which the identity of an entity has been represented.**

Keywords- object-relational; impedance mismatch; ORIM; silo; equivalence

I. INTRODUCTION

Object and relational technologies have proven popular for the development, respectively, of applications and databases, but there are problems that occur when we attempt to combine them in a software system. Each such problem is commonly referred to as an object-relational impedance mismatch (ORIM) [1].

In [2] we explore problems of an ORIM and conclude that there are four kinds of mismatch (conceptual, representation, emphasis and instance), each reflecting a different abstraction (respectively, concept, language, schema and instance). Our framework (Figure 1) recognises two collections of concepts, each provides the basis for a *silo*. A silo comprises artefacts from an abstraction at each level of our framework. The object silo is the left side of Figure 1 and the relational silo is the right side. A level provides a context for the level below.

Our framework highlights that an object and a relational artefact are based on different conceptual frameworks. At the language level an artefact in a silo is described using a particular language. This language is different between silos, for example Java may be used in the object silo and SQL-92 in the relational silo. At the schema level an artefact is created based on a particular design objective. These objectives differ between silos. For example, the design of a program may focus on efficient processing whereas the design of a database may focus on an efficient data structure.

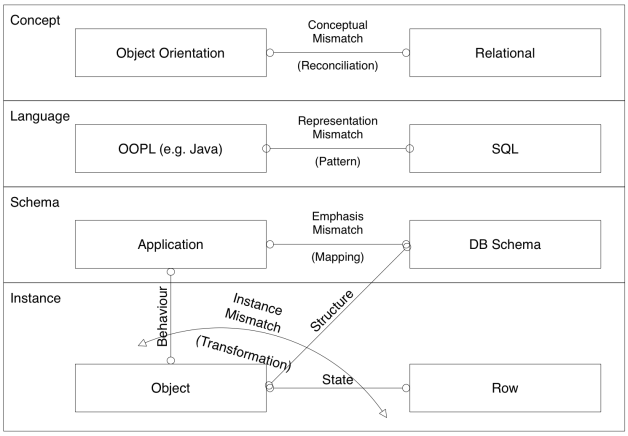

Figure 1. Our Conceptual Framework[1]

During the development of an object-relational application, we combine technologies that make use of these artefacts because each is suited to a particular role. In this paper we highlight that problems of an ORIM arise not just because of differences in language or design objective, but because the semantics and data of an object and a relational artefact are not equivalent.

We develop the idea of equivalence in the context of ORIM and our framework. We provide an example based on the concept of identity and find that there is very little in common between object and relational artefacts. We show that current pattern-based strategies to map identity between object and relational artefacts (e.g. Blaha [3], p420 and Keller [4], p21) have focused on mapping the wrong things. They draw a correspondence between two identity systems but these serve to identify different things. We show that a correspondence should be made between the ways the identity of an entity from a universe of discourse has been modelled, because such an identity is common to both representations. Finally we explore the consequences of equivalence in terms of our framework and propose a new silo.

The paper is structured as follows. Section II sets the context for our work. In Section III we describe the

1 We have chosen to use the label concept rather than paradigm because we understand that a paradigm underpins a conceptual framework Object and relational are the names of two conceptual frameworks.

 ISBN:978-1-61208-115-1

schema level of our framework. In Section IV we begin our exploration of equivalence. In Section V we explain our novel idea of equivalence and introduce an equivalence diagram as a mechanism for exploring a problem of an ORIM. In Section VI we provide an example based on a small case study. In Section VII we explore the options for describing an entity and in Section VIII we highlight the consequences of equivalence for our framework. We present a summary and our conclusions in Section IX and describe future work in Section X.

II. PROBLEMS OF AN ORIM

In [2] we catalogued a number of different kinds of problem of an ORIM. These problems arise because there are differences between the artefacts in each silo. Differences are those of data representation, language syntax and semantics, design approach and conceptual framework.

Object-relational mapping (ORM) strategies have been developed to overcome these differences ([3], [4], and [5]). Each such strategy is based on a correspondence between artefacts in the two technologies. At the language level for example, the definition of a class is used as the basis for the definition of a table.

The rationale for such a correspondence may be that artefacts in different silos appear to be the same abstraction because they have the same name. For example we can name a table ORDER and a class ORDER, or a column QUANTITY and an attribute QUANTITY. Using the same name for two artefacts may appear to endow them with the same semantics, but this is a correspondence that is not justified because they are different abstractions.

A strategy may arise because of a perceived need to represent all of the semantics of an object model in a relational database. One such example is the representation of the semantics of a class hierarchy using SQL-92 (a language that has no such explicit semantics [6]). However using SQL-92, it is not necessary to represent all the semantics of a class hierarchy in order to realise the benefits of a class hierarchy in the design of a relational database.

We argue that a singular focus on correspondence between language artefacts is incorrect. The focus should be on the data and semantics of that which is being represented. In particular the way these data and semantics have been represented using those artefacts.

III. THE SCHEMA LEVEL

We start at the schema level of our framework because it relates directly to the work of those involved in the design of an object-relational application. At this level we are concerned with design artefacts that comprise respectively an object-oriented application and a relational database. We consider the design of each to be a form of schema.

At the schema level of our framework, an object model and a relational model describe aspects of a universe of discourse ([7], p2-1). Whilst a schema uses a particular conceptual framework, language and structure(s) to describe that universe, each schema is a partial representation of the same universe. A universe of discourse therefore provides a point of reference common to both an object and a relational schema. These schemata must be equivalent descriptions of that universe, if we are not to lose information (data and semantics) in a round-trip transformation between an object-oriented application and a relational database. In the next few sections we explore what we mean by an equivalent description at the schema level of our framework.

IV. TWO REPRESENTATIONS OF AN ENTITY

The design of an object-relational application comprises two schemata: one based on the concept of an object and the other on the concept of a relation. Each schema is an abstraction of the same universe of discourse because it is part of the same system. Each schema is also a correct and valid representation of that universe.

The two schemata are also different. Each schema should be based on a collection of concepts, phrased in a particular language and influenced by a design objective. We make the distinction between the formal prescriptive nature of the concepts that underpin the relational model and the relatively descriptive nature of those that underpin an object model. An SQL-92 schema is prescriptive insofar as its language dictates the form of structure into which a representation must fit. An object schema is relatively more descriptive because the semantics and structure of a class are not prescribed in the same way as those of a table. A different person may also produce and therefore influence each schema ([8], p111).

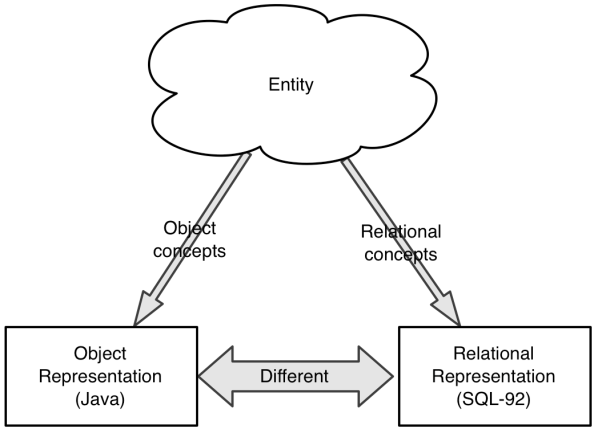

Figure 2. Two Representations of an Entity (type) at the Schema Level

Figure 2 shows an object and a relational representation of the same entity (or entity type) at the schema level. The object representation is formed using artefacts from the Java language. The relational representation is formed using artefacts from SQL-92. We assume that each representation is as complete a representation of the data and semantics of an entity as are possible within a silo.

An entity forms part of a universe of discourse and the description of its data and semantics provides a common

point of reference for both representations. An entity may also be understood as a generalisation of an object and a relational representation. It represents the data and semantics of some thing from a universe of discourse with which we may compare an object and a relational representation.

V. EQUIVALENCE

We consider two representations to be equivalent if they each describe both the same data and the same semantics of an entity. Only those data and semantics that are equivalent can form part of a non-loss transformation between an object and a relational schema. If all the data and semantics of an entity are described in both an object and a relational schema, then none of the data and semantics of that entity should be lost in a round-trip transfer between an application and a database. There may still be differences but these should not impact on the representation of data or semantics. Where there are data or semantics that cannot be preserved in a round-trip transformation between an application and a database, then one schema is able to describe more (or a different subset) of the data or semantics of an entity than the other. Such differences at the schema level are the essence of the kind of object-relational impedance mismatch we label an emphasis mismatch [2].

In both an object and a relational schema one or more artefacts may be used to describe an entity. We use an equivalence diagram to explore differences in the data and semantics of an entity as represented by these artefacts.

An equivalence diagram embodies our notion of equivalence and focuses attention on the essential aspect of Figure 2: that each schema is a description of the same entity. By equivalence at the schema level we mean equivalent descriptions of the same data and semantics of an entity from a universe of discourse.

An equivalence diagram is a Venn diagram comprising two sets. Each set contains the semantics and data of artefacts used to describe an entity in a particular representation. The intersection of these two sets is those data and semantics of an entity that are captured in both representations. These data and semantics will be preserved in a round-trip between an object-oriented application and a relational database.

In Figure 3 the semantics and data of an entity embodied in artefacts used in a schema are represented by a set, drawn as an ellipse. We show two sets: object and relational. The intersection of the two sets represents the data and semantics of an entity common to both schemas. These data and semantics need not be represented in the same way but they are equivalent both to each other and to an idealised representation of entity.

We can use an equivalence diagram in two ways. In the first, we can use equivalence to explore differences of data and semantics between two representations of an entity. We can ask what data and semantics of an entity can be preserved in a round-trip transition from one representation to another. We provide an example in the following sections.

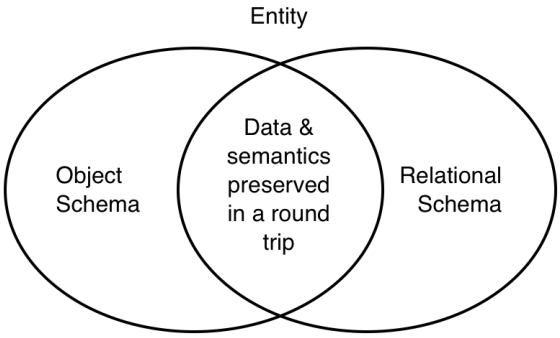

Figure 3. Equivalence between an Object and a Relational representation of an Entity at the Schema Level

In the second, we can use equivalence to improve an ORM strategy. At each level of our framework we can explore the different ways in which the data and semantics of an entity are described by artefacts. At the schema level we consider secondary the artefacts used for the representation of data and semantics of an entity. We describe the consequences for our framework of this use of equivalence in Section VIII.

VI. AN EXAMPLE

In this section we provide an example of the use of an equivalence diagram to explore the identity problem [2]: how do we uniquely identify a collection of data values across both an object and a relational representation?

Figure 4 presents an entity Equity taken from a universe of discourse based on an investment bank. Equity is a particular financial instrument that represents a share in a company.

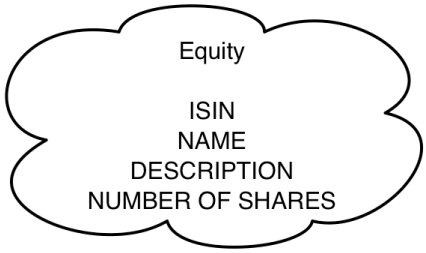

Figure 4. The entity Equity.

An equity is identified by an International Securities Identifying Number (ISIN) code. The ISIN code is defined under ISO 6166 and is unique across all financial instruments. The other attributes are self-explanatory.

From Figure 4 we produce an outline class definition shown in Figure 5 and an outline SQL-92 table definition in Figure 6.

An object ID (OID) is implicit and represents the identity of an object. In Java, for example, it is not necessary to define the OID in the definition of the class of which an object is an instance. Hence, there is no mention of an object ID in the definition of class `Equity` in Figure 5.

```
… class Equity
{       … ISIN;
        … NAME;
        … DESCRIPTION;
        … NUMBER OF SHARES; }
```

Figure 5. The outline of a Java class Equity derived from the entity Equity

The value of an OID is independent of the value of any of the attributes of an object. For example, although an ISIN is unique in the universe of discourse, the OID of an object of class `Equity` will not be based on the value of the attribute ISIN.

```
create table EQUITY(
   ISIN … PRIMARY KEY,
   NAME …,
   DESCRIPTION …,
   NUMBER_OF_SHARES …)
```

Figure 6. The outline of an SQL-92 table derived from the entity Equity

An OID is unique within the execution space of an object-oriented application. An OID guarantees the uniqueness of an object. Two objects with exactly the same attribute values are different objects if they have a different OID. The identity of an object remains the same regardless of any changes to the value of its attributes. For example, two objects of class `Equity` are different even if they have the same value for the attribute ISIN. In order to prevent this erroneous situation, a constraint must be implemented in a method. Furthermore, changing the value of the attribute ISIN of an object does not change the value of its OID.

The identity of a tuple is the value of all its domains. As such the identity of a tuple is dependent on the value of a domain. In a single relation there cannot be two tuples with the same value for each domain. A primary key of a relation is not necessary for identity but does provide a short-form of reference to a tuple.

At the language level, the semantics of a table are different. A duplicate row is permissible. A primary key enforces uniqueness of a row in a table and restricts the identity of a row to those columns in a key. For example ISIN is the primary key of table EQUITY. There cannot be two rows in this table with the same value for this column. The value of a primary key column in a row should not be changed because this affects the identity of that row.

In Figure 7 we provide an example of an equivalence diagram for the semantics of identity in an object and a relational schema. This shows that there is little in common between the object and relational semantics of identity at the schema level. A row and an object are not the same thing. An OID and the primary key ISIN have little correspondence. The only semantics they share is that each uses identity as a mechanism for ensuring an occurrence is distinct.

These differences are realised at the language level of our framework and above. An OID is not a building block of an object framework, rather it is a programming necessity introduced at the language level. At the concept level an object is distinct so there is no need for an OID. Similarly, at the language level a primary key uniquely identifies a row in a table. A tuple is distinct by definition. At the concept level therefore we have an object and a tuple, each is unique but they have no common basis for this uniqueness. We have used the levels of our framework to pinpoint the cause of the identity problem. We should not attempt a correspondence between an OID and a primary key because they have no common basis for uniqueness.

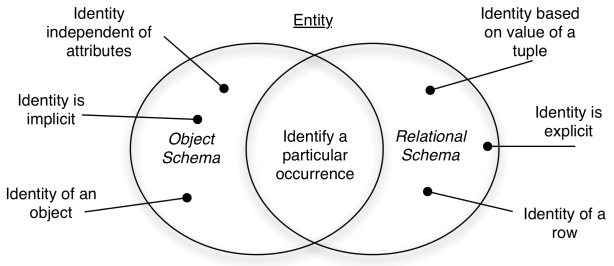

Figure 7. Exploring Identity Between Object and Relational Representations of an Entity

Pattern strategies using SQL-92 (e.g. Blaha [3] and p420, Keller [4], p21) map the semantics of identity between the identity systems employed in an object and a relational schema. For example, they suggest we should introduce a new column into the table EQUITY. This column would store the value of an identity of an object of class `Equity`. Keller [4] suggests using an application-generated identity they call a synthetic identity rather than the actual value of an object ID. Even in our simple example this strategy has shortcomings.

An OID is unique only within the execution space of a single object-oriented program. The correspondence between an ISIN and an OID is only temporary. An OID cannot be used as a primary key because it is not guaranteed to be unique in a database. Even if we extract the value of an ID from an object, that value has no meaning in a database. It also has no semantics in the universe of discourse from which a database schema is derived. A synthetic object ID may somehow be unique across executions but an object and a tuple are different abstractions and such an ID also has no meaning in a universe of discourse. An object ID does not have the semantics necessary to be used as a primary key in a database.

Using equivalence we understand that a mapping between representations at the schema level should be based on correspondence between the mechanisms used to describe the identity of an entity. This mechanism may not be the same as the identity used in each identity system but the identity of an entity is common to both representations. In our example we should make a correspondence between an object and a relational representation of the attribute ISIN of entity Equity, because this is the identity of the entity Equity and it is common to both representations. We

should not make a correspondence between an object ID and a primary key because these identify different things. A consequence of making a correspondence between these identity systems is that a transformation strategy is then required to form a correspondence between identity values.

VII. DESCRIBING AN ENTITY

The language for describing an entity is an important choice. A description of an entity cannot favour one of the two conceptual frameworks. This would limit the description of the semantics of an entity to those that could be expressed using just one of the frameworks. How then do we describe an entity without favouring one conceptual framework over another?

Dieste [9] describe what they term a generic conceptual model (GCM). The objective of a GCM is to describe knowledge of a requirement in a way that does not determine (what they refer to as) the implementation paradigm. They provide a number of transformations of a GCM into a target conceptual model in a particular implementation paradigm. Whilst the term paradigm was originally intended to describe the set of practices that define a scientific discipline at any particular period of time [10], it has been used in computing as a label for a particular viewpoint. We understand that a paradigm underpins a conceptual framework, and that object and relational are two conceptual frameworks.

The approach of Dieste is set within a software development lifecycle. The objective of a GCM is to delay commitment by providing a description of a universe of discourse independent of an implementation paradigm. Once a choice of implementation paradigm has been made, a GCM is transformed into a model based on a particular collection of conceptual building blocks. A GCM is therefore independent of both an object and a relational conceptual framework.

The language employed in the production of a GCM is a possible candidate for the description of an entity. Unlike an element of a GCM, an entity is not used as the basis for the generation of a representation in a particular conceptual framework. Rather a description of the semantics and data of an entity may be used as the basis for exploring equivalence.

Multi-paradigm Modelling (MPM) is another area in which we may find a candidate for the description of an entity. Multi-paradigm modelling is concerned with "developing a set of concepts and tools to address the challenge of integrating models of different aspects of a software system specified using different formalisms and eventually at different levels of abstraction" [11]. Integrating heterogeneous models is one of the most important challenges of MPM. Amaral [11] notes that "the topic on model composition is of very high interest but one that raises a number of very difficult issues". Various authors (Jiang et al., Yie et al. and Barroca et al. in [11], p222) have explored dependencies between models, model transformations and language composition. Our framework provides a means to structure an exploration of these issues. The issue of dependency between models occurs at the schema level of our framework whilst issues of language composition occur at the language level of our framework. Those working in the area of MPM will benefit from our understanding of equivalence because equivalence is essential for the preservation of semantics between models.

VIII. EQUIVALENCE AND OUR FRAMEWORK

We have explored equivalence at the schema level and shown that it may be possible to produce a description of an entity independent of an object and a relational conceptual framework. The concept of an entity is only relevant at the schema level because a schema is a representation of a universe of discourse. In this section we explore the consequences of equivalence in the context of our framework and explain the basis for equivalence at the other levels.

The concept level of our framework provides the context for the language level that in turn provides the context for the schema level. We can use this contextualisation to reflect on the description of an entity at the schema level. The example of identity has highlighted for example, that issues of language influence the semantics of an entity as described in a schema.

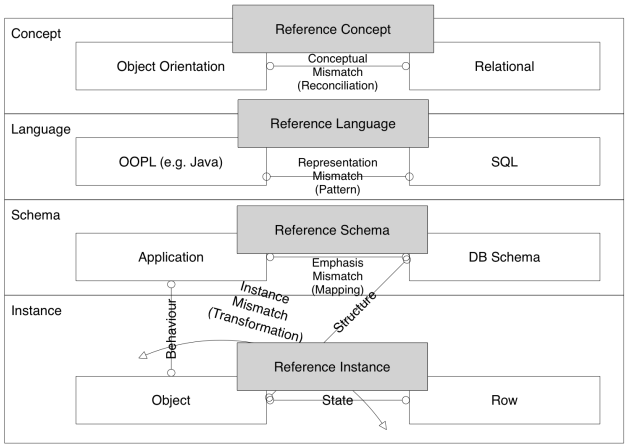

Figure 8. Our Conceptual Framework including the Reference Silo

The description of an entity may be viewed as a generalisation of an object and a relational description. The description of an entity must be phrased in terms of a language that is itself a generalisation of an object and a relational language. The language used to describe an element map ([9], Section 3.1) provides a possible candidate. A conceptual framework that is a generalisation of an object and a relational conceptual framework will underpin the language. We therefore propose a third silo in our framework and we label this the reference silo.

The reference silo (shown down the centre of Figure 8) is currently theoretical and artefacts within it an ideal, but its purpose can be related to the work we describe in Section VII. In this silo there is a reference concept level, a reference language level, a reference schema level and a

reference instance level. Each level provides artefacts for, or influences the description of an entity from a universe of discourse within a reference schema. This description does not need to be perfect, but as a minimum it must be a generalisation of those data and semantics that may be described using an object and a relational artefact.

At each level of our framework within the reference silo we can explore equivalence. We can explore equivalence between the data and semantics of a reference artefact and those data and semantics described in an object and a relational artefact. The data and semantics of a reference artefact described by an object or a relational artefact are shown as a set in an equivalence diagram. Depending on the level of the framework, that set may contain conceptual building blocks, language structures, design representations or data formats.

TABLE I. SOME BUILDING BLOCKS OF THE OBJECT AND THE RELATIONAL CONCEPTUAL FRAMEWORKS

Conceptual Framework	Building Blocks
Object [12]	Object, Class, Association, Method, Attribute, Subclass
Relational [13]	Relation, n-tuple, Domain, Column, Projection, Join, Restriction, Composition, Primary Key

At the concept level of our framework for example, a set comprises the building blocks employed by a conceptual framework. Table I provides an example of some of the building blocks employed by the object and the relational conceptual frameworks. The intersection of the two sets comprises those data and semantics of a reference artefact that are represented by artefacts in both the object and the relational silo.

IX. SUMMARY AND CONCLUSIONS

Problems of an ORIM exist not just because artefacts are described using a different language, but also because an object and a relational representation are based on different conceptual frameworks. This distinction underpins our conceptual framework.

A conceptual framework underpins the language, schema and data used to describe an entity from a universe of discourse. If we are to preserve the data and semantics of an entity from a universe of discourse in a round-trip between an object-oriented application and a relational database, the description of that entity in each schema must be equivalent.

The novel perspective of equivalence facilitates an understanding of an impedance mismatch between an object and a relational artefact. We found at the schema level there is little in common between the semantics of an object and a relational system of identity. ORM strategies have failed to recognise this and instead make a correspondence between identity systems. In order to avoid problems of an ORIM, the correspondence implicit in an ORM strategy should be based on how the data and semantics of the identity of an entity are described in each representation.

Equivalence is not only a schema level concern involving the description of an entity. Reflecting on the contextualisation provided by our framework we introduced the reference silo. This silo comprises the artefacts used to describe an entity at each abstraction.

At each level of our framework we can explore equivalence between an artefact in the reference silo and those in the object and relational silos. Such an exploration will provide further insights into the most appropriate way to address problems of an ORIM.

Whilst the reference silo is still an ideal, we note that there is work in the areas of a GCM and MPM that may lead to the realisation of artefacts in this silo. Our framework will also help those working in the area of MPM.

X. FUTURE WORK

Our technique of equivalence may be used to explore other problems of an ORIM [2]. Such an exploration will demonstrate further our technique, and may result in improvements to other ORM strategies. Finally, further work is required to understand the contribution of our framework and the technique of equivalence, to MPM and an exploration of the issues identified by Amaral [11].

REFERENCES

[1] G. Copeland and D. Maier: Making Smalltalk a database system. ACM SIGMOD Record 14 (1984) 316-325

[2] C. Ireland, D. Bowers, M. Newton and K. Waugh: A Classification of Object-Relational Impedance Mismatch. In: Chen, Q., Cuzzocrea, A., Hara, T., Hunt, E., Popescu, M. (eds.): The First International Conference on Advances in Databases, Knowledge and Data Applications, Vol. 1. IEEE Computer Society, Cancun, Mexico (2009) p36-43

[3] M.R. Blaha, W.J. Premerlani and J.E. Rumbaugh: Relational database design using an object-oriented methodology. Communications of the ACM 31 (1988) 414-427

[4] W. Keller: Mapping Objects to Tables: A Pattern Language. In: Bushman, F., Riehle, D. (eds.): European Conference on Pattern Languages of Programming Conference (EuroPLoP), Irsee, Germany (1997)

[5] M.L. Fussell: Foundations of Object Relational Mapping (http://www.chimu.com/publications/objectRelational/) (Accessed: 25th September 2007)

[6] C. Ireland, D. Bowers, M. Newton and K. Waugh: Understanding Object-Relational Mapping: A Framework Based Approach. International Journal On Advances in Software **2** (2009)

[7] J.J.v. Griethuysen (ed.): Concepts and Terminology for the Conceptual Schema and the Information Base. ISO, New York (1982)

[8] S.W. Ambler: Agile Database Techniques - Effective Strategies for the Agile Software Developer. Wiley (2003)

[9] O. Dieste, M. Genero, N. Juristo, J.L. Mate and A.M. Moreno: A conceptual model completely independent of the implementation paradigm. The Journal of Systems and Software 68 (2003) 183-198

[10] T.S. Khun: The Structure of Scientific Revolutions. The University of Chicago Press, Chicago, IL (1970)

[11] V. Amaral, C. Hardebolle, G. Karsai, L. Lengyel and T. Levendovszky: Recent Advances in Multi-paradigm Modeling. Models in Software Engineering, Vol. 6002/2010. Springer-Verlag, Berlin (2010) 220-224

[12] J. Rumbaugh, I. Jacobson and G. Booch: The Unified Modeling Language Reference Manual. Addison Wesley (2005)

[13] E.F. Codd: A relational model of data for large shared data banks. Communications of the ACM 13 (1970) 377-387

 ISBN:978-1-61208-115-1