Evolving Temporal Conceptual Schemas: The Reification Case

Cristina Gómez[1], Juan-Ramón López[1,2], Antoni Olivé[1]
[1]*Universitat Politècnica de Catalunya, Departament de LSI.*
Barcelona - Catalonia.
[2]*Universidade da Coruña, Departamento de Computación*
{cristina|jrlopez|olive}@lsi.upc.es

Abstract

In this paper we study temporal conceptual schema evolutions related to reification, a typical and complex modeling construct. Various types of reification are considered. Using a previously defined framework, we specify, only at conceptual level (and without descending to logical or application levels), the effects of any possible evolution related to reification, thus reducing the complexity of the management of those changes.

1. Introduction

This paper deals with the evolution of temporal conceptual schemas of information systems. Specifically, we are interested in the reification case. Reification (also known as *associative entity*, *association type*, *association class* or *relationship-to-entity*) is a construct that is commonly used in the conceptualization of information systems.

Ideally, evolution of information systems should be managed at conceptual schema level, and automatically propagated down to the logical schema(s) and application programs [4]. Unfortunately, existing technology is far from allowing this ideal situation, and evolution must be managed also at design and application levels, thus incrementing its complexity. Database schema evolution is a widely investigated field [1, 15, 12]), but few research efforts have been done in the (temporal) conceptual schemas evolution topic [14, 7, 6, 3]. Our work pretends to contribute partially to an improvement in this area, focusing our efforts in the reification construct and in its different types [11]. We completely specify, only at conceptual level, any possible evolution related to reification, extending a previously defined framework [8] that supports schema evolutions related to some common conceptual modeling elements.

The rest of this paper is organized as follows: We review the framework and its main features in Section 2, and reification definition and its types in Section 3. Our contribution is presented in Section 4, which deals with reification schema evolutions. Finally, the conclusions of the paper are presented in Section 5.

2. Framework Review

The framework is based on the well-known three-level architecture of an information system [5], which is seen as the union of a conceptual schema (for short, schema), an information processor (IP) and an information base (IB). The IB is a representation of the state of the information system domain, whose structure is shaped by the schema. The IP receives messages from the information system environment, modifies the IB, if necessary, and sends messages to the environment as a response, if needed.

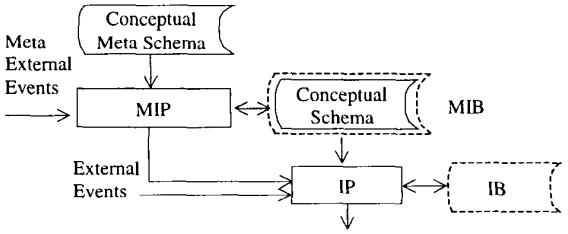

Figure 1. Framework for the evolution of temporal conceptual schemas

To deal with evolution, a reflective approach is followed [9]. Temporal schemas are considered as the domain of a special *meta* information system. The framework (Figure 1) has two levels. The lowest level corresponds to the conventional information system (conceptual schema, IP, IB). The highest level corresponds to the meta information system: there is a conceptual meta schema, a meta information processor (MIP) and a meta information base (MIB). There is a one to one correspondence between the MIB and the schema. The only way of evolving the schema is by modifying the MIB: different schema evolution operations are

0-7695-0906-1/01 $10.00 © 2001 IEEE

represented by meta external event types. When occurrences of those event types are produced, the MIP changes appropriately and, therefore, the schema changes too. The MIP can also generate external events and notify the IP of them, if changes in the IB are also necessary as a consequence of the schema change.

For the following descriptions, we use the first order logic (FOL) as modeling language. However, we also use the UML notation in some figures, for illustration purposes only. In our temporal view, time is assumed as linear, discrete and totally ordered [13], and time points are expressed in a common base unit, such as second, minute, or day.

2.1 Conceptual schemas

The framework pretends to be general, and not linked to a specific conceptual modeling language. For that reason, only a few modeling elements, which are common to most languages and enough for the definition of any possible schema, are taken into account. Here we present a short description of those elements. A more detailed description can be found in [8].

Objects of the information system domain are represented in the IB as instances of *entity* and *relationship types*. Changes in the IB are produced by *event types* instances, and restricted by some *integrity constraints*. *Derivation* rules can be defined to automatically settle the instances of any particular entity, relationship or external event type. All of these elements are represented in the schema and the IB using logical formulas. For instance, with predicate $R(e_1,...,e_n,t)$ we represent in the IB the fact that $(e_1,...,e_n)$ is instance of an n-ary ($n \geq 2$) relationship type R at time t, being $e_1,...,e_n$ the participant entities in the relationship.

Each one of these five elements has its own additional properties [11]. For instance, the *durability* property for entity types can take two values: we say that an entity type E is *instantaneous* if for any instance e and time t, if we have $E(e,t)$ then $E(e,t+1)$ does not hold or e is instance of E only at some subintervals of t. In any other case, E is *durable*. The definition is similar for relationship and external event types.

External event types have also related features: an event type definition includes the *parameters* of the event, its *preconditions* and its *effect rules*. The effect rules are formulas that define the consequences of the event occurrence over the IB, by generating instances of *structural event types* [10], which are implicitly defined in the schema. There is an insertion and a deletion structural event type for each entity and relationship type. For instance, given an n-ary relationship type R, the induced effect in the IB of an event $Ins\_R((e_1,...,e_n),t)$ is that the relationship $(e_1,...,e_n)$ becomes an instance of R at t. The induced effect in the IB of an event $Del\_R((e_1,...,e_n),t)$ is that the relationship $(e_1,...,e_n)$ ceases to be an instance of R at t. The same applies to structural event types corresponding to entity types.

2.2 Minimal Meta Schema

Figure 2 shows, in UML notation, the meta schema which allows the definition in the MIB of any possible schema in terms of the schema core elements. In fact, the meta schema is recursively defined in terms of the five core elements[1], in this case: meta entity types, meta relationship types, meta integrity constraints, meta derivation rules and meta external event types. More details can be found in [8].

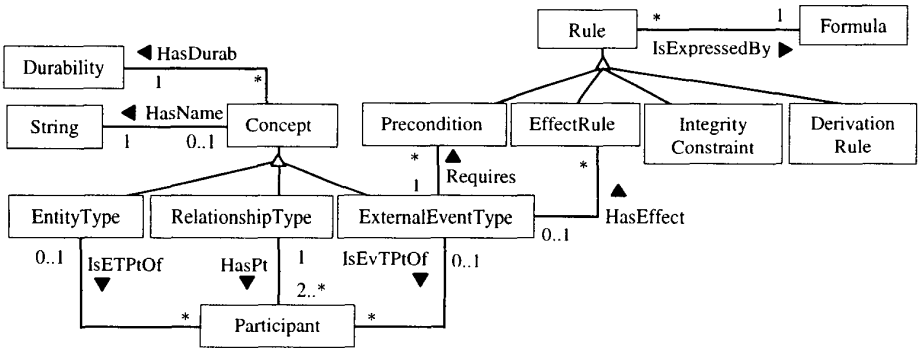

Figure 2. Minimal conceptual meta schema (Simplified)

2.3 Correspondence Rules

The consistency between the MIB and the conceptual schema is achieved by defining a set of *correspondence rules*. These rules state and ensure that changes in the MIB have its counterpart in the schema. The (simplified) rule corresponding to entity types is the following:

> ***CR1.*** *There is a one-to-one correspondence between the instances at time* t *of* EntityType *in the MIB and the entity types in the schema. Instances of* EntityType *have, at* t, *the durability and the name assigned through* HasDurab *and* HasName *at* t.

Additional detailed rules corresponding to relationship types, integrity constraints, derivation rules and external event types can also be found in [8].

[1] Figure 2 includes, for clarity, additional modeling elements as partitions. Really, in the meta schema, those partitions are expressed in terms of derivation rules and integrity constraints.

2.4 Meta Schema Extensions

It is possible to define meta schema extensions to support other constructs used in conceptual modeling (e.g. cardinality constraints, partitions), in order to ease the designer's task when defining a conceptual schema.

Each meta schema extension shapes the structure of the representation, in the MIB, of instances of a construct. The extension must also include, expressly, some additional meta derivation rules. These rules translate MIB representations of the construct into minimal meta schema instances. This way, the correspondence rules of Section 2.3 are not affected and remain simple and completely effective. Some detailed examples can be found in [8].

3. Reification Review

In this paper we assume the reification types definition given in [11], and we reproduce it below for ease of reference.

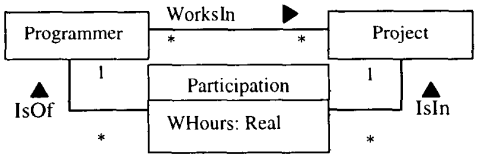

Figure 3. Reification result

Informally, the reification transformation consists in considering a relationship type as an entity type, in order to allow the use of the reified entity type as a participant in other relationship types. Formally, reification of a n-ary relationship type R is a schema transformation that produces a new *intrinsic* entity type E and n *intrinsic* relationship types $R_1...R_n$ which connect E with the n participant entity types in R.

Figure 3 shows an example of the result of reification. A relationship type *WorksIn* between *Programmer* and *Project* is reified into a *Participation* intrinsic entity type and two intrinsic binary relationship types, *IsOf* and *IsIn*. The intrinsic entity type participates in a binary relationship type (or attribute) called *WHours* which represents a number of working hours.

From a temporal perspective, various different types of reification can be defined, and they can coexist for the same relationship type. The example includes only one reification, and its semantics differ depending on the reification type considered: reification per instant, reification per classification interval and reification per life span.

Figure 4 shows the semantics of the three reification types considered for the example showed in Figure 3.

4. Reification Evolution

After the necessary background definitions and reviews, we can center now our attention into the real motivation of this paper: to deal with conceptual schema evolutions associated to reification.

Some possible elementary schema changes related to the reification construct are the definition and the removal of a reification of a relationship type. Moreover, changes related to other constructs, like removal and modify the frequency and durability of a reified relationship type, can affect indirectly to reifications.

4.1 Reification Meta Schema Extension

We want to specify the effects of the possible changes at conceptual level, without descending to the design or application level. Therefore, we apply the framework just reviewed to the reification case.

To define an extension of the meta schema to support reification, we assume as derived concepts the intrinsic types that result from a reification. The intrinsic entity type and the intrinsic relationship types will be populated in the IB by a schema derivation rule, which will infer their instances from the instances of the reified relationship type.

Figure 5 shows the conceptualization of reification in a meta schema extension. It includes some new meta entity and relationship types. We allow the reification of any *RelationshipType*. Reification must have a *ReificationType*. We consider useful to partition meta entity types *EntityType* and *RelationshipType* into an intrinsic and a non-intrinsic subtype. The instances of *IntrinsicRelationshipType, IntrinsicEntityType* will be the (derived) schema intrinsic types generated as a consequence of reification.

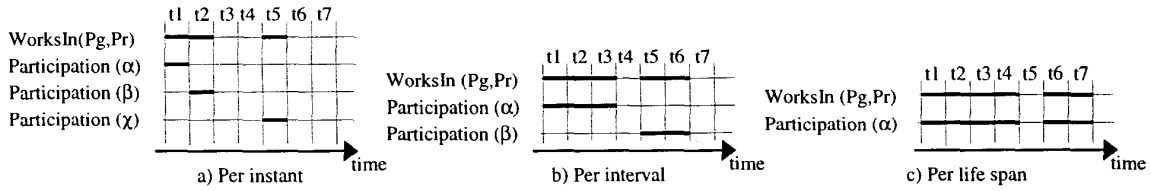

Figure 4. Examples of different reification types

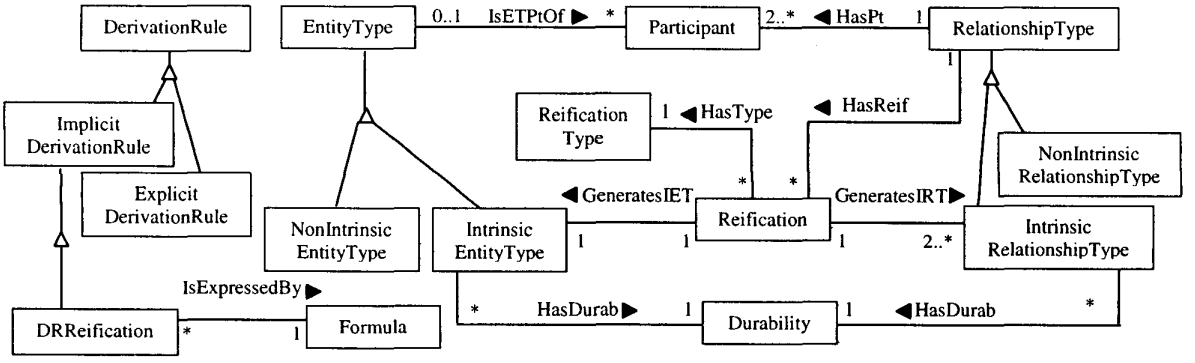

Figure 5. Meta schema conceptualization of reification

Extending the meta schema requires also several meta integrity constraints and meta derivation rules. Due to space limitations, we show here only an example of each type of rules. A meta integrity constraint which states that only reifications can generate intrinsic entity types is the following:

MIC: *GeneratesIET(re,ie,t)* $\rightarrow$ *Reification(re,t)* $\wedge$ *IntrinsicEntityType(ie,t)*

An example of a meta derivation rule, which automatically defines the correct durability property of the intrinsic types, is:

MDR: *HasDurab(ie,d,t)* $\leftarrow$ *GeneratesIET(re,ie,t)* $\wedge$ *HasType(re,ty,t)* $\wedge$ *HasReif(r,re,t)* $\wedge$ *((ty=Instant* $\wedge$ *d=Instantaneous)* $\vee$ *((ty=Interval* $\vee$ *ty=Life)* $\wedge$ *HasDurab(r,d,t)))*

Other MDR's allow defining automatically the correct cardinalities of the intrinsic relationship types, or the durability and the schema derivation rules (*DRReification*) corresponding to each intrinsic type. Here we can show an example of these rules for the reification per interval case:[2]

If the reified relationship type *R* is instantaneous:

"E(e,t) $\wedge$ $R_1(e,e_1,t) \wedge \ldots \wedge R_n(e,e_n,t) \leftarrow R(e_1,\ldots,e_n,t)$ $\wedge$ *e=newSymbol()"*

If *R* is durable:

$"E(e,t) \wedge R_1(e,e_1,t) \wedge \ldots \wedge R_n(e,e_n,t) \leftarrow R(e_1, \ldots, e_n,t) \wedge$
$((E(e,t\text{-}1) \wedge R_1(e,e_1,t\text{-}1) \wedge \ldots \wedge R_n(e,e_n,t\text{-}1))$
$\vee (\neg R(e_1, \ldots, e_n,t\text{-}1) \wedge e=newSymbol()))"$

The rest of meta integrity constraints and meta derivation rules can be found in [2].

4.2 Meta External Event Type: M_AddReification

As we saw in Section 2, schema evolution is performed in the framework by meta external events. When a designer wants to do a change in the schema, she generates an instance of the appropriate meta external event type. The corresponding effect rules will induce meta structural events, which change the MIB. By the correspondence rules, such changes imply changes in the schema, and maybe in the IB. We present here the meta external event type which allows to define reifications, *M_AddReification(rer,t)*. Other related meta external event types are described in [2].

M_AddReification(rer,t) has three parameters: the relationship type *r* that is going to be reified, the *name* of the new entity type, and the *type* of the reification.

Preconditions. It cannot exist another reification of the same type for the considered relationship type.

Effect rules. A first set of effect rules induces the meta structural events that create a new intrinsic entity type *ie*, with the name given as an event parameter.

ie=newSymbol()
M_Ins_IntrinsicEntityType(ie,t)
M_Ins_HasName(ie,name,t)

Now we must create a new reification *re*, linked to its corresponding relationship type *r* and the new entity type *ie* in which *r* is going to be reified. The reification type is also established.

re=newSymbol()
M_Ins_Reification(re,t)
M_Ins_GeneratesIET(re,ie,t)
M_Ins_HasReif(r,re,t)
M_Ins_HasType(re,type,t)

Finally, for each participant entity type *ei* of *r*, a new intrinsic relationship type *ri* must be created, linking the new intrinsic entity type *ie* and *ei*. The correct cardinality constraints must be established for both participants in each *ri*. A system-generated name is assigned to the new intrinsic relationship types. If the designer wants to change these names, she can do it later. The following effect rule works for participants that are entity types:[3]

M_Ins_IntrinsicRelationshipType(ir,t) $\wedge$
M_Ins_HasName(ir,ir,t) $\wedge$ *M_Ins_Participant(pi',t)* $\wedge$
M_Ins_HasPt(ir,pi',t) $\wedge$ *M_Ins_IsETPtOf(ie,pi',t)* $\wedge$
M_Ins_Participant(pi'',t) $\wedge$ *M_Ins_HasPt(ir,pi'',t)* $\wedge$
M_Ins_IsETPtOf(ei,pi'',t) $\wedge$ *M_Ins_HasMinCard(pi',1,t)* $\wedge$

[2] Note that, in what follows, we use the special predicate *newSymbol()* to obtain new MIB or IB symbols.

[3] *HasMinCard* and *HasMaxCard* are meta relationship types defined in the minimal meta schema extension to support cardinalities, and later translated into common integrity constraints using meta derivation rules.

M_Ins_HasMaxCard(pi',1,t)
← ir=newSymbol() ∧ pi'=newSymbol() ∧
pi''=newSymbol() ∧ HasPt (r,pi,t-1) ∧
IsETPtOf(ei,pi,t-1)

An analogous effect rule is needed for participants that are event types.

Induced effects on the MIB. The above effect rules are the only ones that need to be defined. We now reason (informally) about induced effects on the MIB by MDR's defined in the minimal meta schema and its extensions: *ie* becomes instance of *EntityType* and *Concept*, and $ir_1,...,ir_n$ begin to be instances of *RelationshipType* and *Concept*. *Reification re* is automatically associated (by a MDR not shown here) to each *IntrinsicRelationshipType* (i.e. $ir_1,...,ir_n$) through some instances of *GeneratesIRT*. Each $ie,ir_1,...,ir_n$ *HasDurab* at *t*. The *Formula* in which *IsExpressedBy* the required *DRReification* for the intrinsic types $ie,ir_1,...,ir_n$ is induced. Each instance of *DRReification* becomes instance of *Rule, DerivationRule* and *ImplicitDerivationRule*. The cardinalities defined for the intrinsic types using *HasMinCard* and *HasMaxCard* are translated into a corresponding *IntegrityConstraint* which *IsExpressedBy* a *Formula*.

Induced effects on the schema and IB. By correspondence rules, these are the changes in the schema at t: *ie* is an entity type with the defined durability and name. $ir_1,...,ir_n$ are binary relationship types with the defined durability and cardinalities. New cardinality integrity constraints for the intrinsic relationship types are defined. The *Formula* assigned to each *DRReification* induced is a new derivation rule of the schema, which sets the instances, in the IB, of its corresponding intrinsic type from the instances of the reified relationship type.

5. Conclusions

In this paper we have studied temporal conceptual schema evolutions related to the reification construct. As the temporal dimension has been taken into account, some different kinds of reification have been considered.

To deal with evolution, we have applied a previously defined framework for temporal conceptual schemas evolution. We have seen that, with only some extensions, the framework easily supports reification in all of its types, and is able to manage any potential schema change that involves it. The effects of the possible changes have been specified at conceptual level only, thus increasing the understandably of the specification and reducing the complexity of the management of changes.

Acknowledgments

The authors are indebted to the anonymous referees for their helpful comments. This work has been partly supported by CICYT program projects TIC99-1048-C02-1 and TEL99-0335-C04-2.

References

[1] J. Banerjee, H.T. Chou, J.F. Garza, W. Kim, D. Woelk, N. Ballou, "Data Model Issues for Object-Oriented Applications", *ACM TOIS Vol.5, No.1*, January (1987) pp.3-26

[2] C. Gómez, J.R. López, A. Olivé, "Evolving Temporal Conceptual Schemas: The Reification Case (extended version)". Research Report LSI-00-50-R, LSI, UPC.

[3] I. Goralwalla, D. Szafron, T. Özsu, R. Peters, "A Temporal Approach to Managing Schema Evolution in Object Database Systems", *Data&Knowledge Eng. 28(1)*, October (1998) pp. 73-105

[4] J.L. Hainaut, V. Englebert, J. Henrard, J.M. Hick,D. Roland, "Database Evolution: the DB-MAIN Approach", *13th. Intl. Conf. on the Entity-Relationship Approach - ER'94*, LNCS 881, Springer-Verlag (1994) pp. 112-131

[5] ISO/TC97/SC5/WG3: "Concepts and Terminology for the Conceptual Schema and Information Base". J.J. van Griethuysen (ed.), March (1982)

[6] M. Jarke, R. Gallersdrfer, M.A. Jeusfeld, M. Staudt, S. Eherer, "ConceptBase - a deductive object base for meta data management", *Journal of Intelligent Information Systems, 4(2)* (1995) pp. 167-192

[7] T. Lemke, R. Manthey, "The Schema Evolution Assistant: Tool Description", *IDEA.DE.22.O.004*, University of Bonn.

[8] J.R. López, A. Olivé, "A framework for the evolution of temporal conceptual schemas of information systems", *Proc. Conference in Advanced Information Systems Engineering - CAiSE'00,* LNCS 1789, Springer-Verlag (2000) pp. 369-386. An extended version can be found at: Research Report LSI-00-14-R, LSI, UPC. (2000)

[9] R. Manthey, "Beyond Data Dictionaries: Towards a Reflective Architecture of Intelligent Database Systems", *DOOD'93*, Springer-Verlag (1993) pp. 328-339

[10] A. Olivé, "On the design and implementation of information systems from deductive conceptual models", *Proc. VLDB'89*, Amsterdam, The Netherlands (1989) pp. 3-11

[11] A. Olivé, "Relationship Reification: A temporal View", *Proc. CAiSE'99*, Heidelberg, Germany, LNCS 1626, Springer (1999) pp. 396-410

[12] J.F. Roddick, "Schema Evolution in DataBase Systems – An Updated Bibliography", *ACM SIGMOD Rec., 21(4)*, May (1994) pp. 35-40.

[13] C.I. Theodoulidis, P. Loucopoulos, "The time dimension in conceptual modeling",.*Information Systems, Vol 16, No. 3*, Pergamon Press (1991) pp. 273-300

[14] M. Tresch, M.H. Scholl"Meta Object Management and its Application to Database Evolution", *Proc. 11th. Intl. Conf. on the Entity-Relationship Approach - ER'92*, LNCS 645, Springer-Verlag, pp. 299-321.

[15] R. Zicari, "A Framework for Schema Updates in Object-Oriented Database System", in F. Bancilhon, C. Delobel, P. Kanellakis (ed.) "Building an Object-Oriented Database System - The Story of O_2", Morgan Kaufmann, pp.146-182.